UNIVERSIDADE FEDERAL DE SANTA CATARINA

CURSO DE PÓS GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UM SIMULADOR INTEGRADO PARA LABORATÓRIOS DE COMPUTAÇÃO

DISSERTAÇÃO SUBMETIDA À

UNIVERSIDADE FEDERAL DE SANTA CATARINA

COMO REQUISITO PARCIAL PARA A OBTENÇÃO DO GRAU DE

MESTRE EM CIÊNCIA DA COMPUTAÇÃO

CARLOS BENEDITO SICA DE TOLEDO

FLORIANÓPOLIS, JULHO DE 1994

UM SIMULADOR INTEGRADO PARA LABORATÓRIOS DE COMPUTAÇÃO

Carlos Benedito Sica de Toledo

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA OBTENÇÃO DO TÍTULO DE

MESTRE EM CIÊNCIA DA COMPUTAÇÃO

ESPECIALIDADE EM SISTEMAS DE COMPUTAÇÃO E APROVADA EM SUA FORMA FINAL PELO PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO.

Hermann Adolf Harry Lücke, Dr. Ing.
Orientador e Coordenador do Curso

BANCA EXAMINADORA

Hermann Adolf Harry Lücke, Dr. Ing.
Presidente

Álvaro José Periotto, D.Sc.

Antonio Augusto R. Coelho, Dr.

Luiz Fernando Jacintho Maia, Dr.

Dedico este trabalho

Aos meus queridos pais: Maria de Lourdes Sica de Toledo e José Mariano de Toledo Costa, que sempre me incentivaram ao estudo, e de alguma forma, ainda o fazem.

Vossa Benção.

E com muito amor, à família que estamos construindo: minha esposa Arlete Cristina Borghi e minhas filhas Mariana Borghi Sica de Toledo e Lígia Borghi Sica de Toledo.

AGRADEÇO:

Em primeiro lugar à Deus, à quem eu recorri, sempre que me senti em apuros e à Arlete, Mariana e Lígia, que muitas vezes me serviram arrimo;

Ao Hermann, pela orientação e pela amizade;

À Universidade Estadual de Maringá;

Aos orgãos financiadores de pesquisa CAPES e CNPq;

Ao Mauro Madeira, pelo longos papos sobre orientação a objetos;

Ao Eduardo Kern, pela sua dedicação;

Ao Vicente Crisostomo, pela constante hospedagem;

A Verinha, pela grande dedicação;

E a todos os amigos, pelo mágico poder de nos levar sempre para frente, *valeu galera.*

GLOSSÁRIO

UCP	-	Unidade Central de Processamento
UC	-	Unidade de Controle
PPI	-	Programmable Peripheral Interface
LSB	-	Least Significant Bit
MSB	-	Most Significant Bit
CUA	-	Current User Acess
OOP	-	Object Oriented Paradigm
Abstração	-	Capacidade mental que nos permite visualizar problemas do mundo real, dividindo-os em graus de detalhes dependendo do contexto do problema
Objeto	-	Um conceito, abstração, ou utensílio com limites e princípios bem definidos para um problema; instância de uma classe.
Classe	-	Descrição de um grupo de objetos com propriedades similares, e comportamento, relacionamento e semânticas comuns
Classe abstrata	-	Uma classe que não pode ser instanciada diretamente, porém, deus descendentes podem ter instâncias
Classe abstrata pura	-	Classe abstrata que possui um método puro (sem corpo e inicializado com zero)
Ligação	-	Conexão física ou conceitual entre dois objetos.
Associação	-	Relação entre instâncias de duas ou mais classes, descrevendo um grupo de ligações com estruturas e semânticas comuns.
Agregação		Forma especial de associação, entre um conjunto e suas partes.
Atributo	-	Propriedade (com nome) de uma classe, que descreve um valor de dado diferente para cada objeto de uma classe
Operação	-	Transformação que pode ser aplicado a objetos de uma classe, neste contexto, sinônimo função.
Método	-	Implementação de uma operação para uma classe específica
Público	-	Em referências aos atributos ou métodos de uma classe, é a parte do classe que é acessível por métodos de todas as outras classes
Protegido	-	Em referências aos atributos ou métodos de uma classe, é a parte do classe que é acessível por métodos das classes herdeiras, num nível imediatamente abaixo da classe corrente.
Privado	-	Em referências aos atributos ou métodos de uma classe, é a parte do classe que é acessível somente por métodos da própria classes
Polimorfismo	-	Capacidade de um mesmo método se comportar diferente em classes diferentes
Construtor	-	Método que inicializa uma nova instância de classe.
Construtor	-	Método que retira um objeto instanciado da memória
Encapsulamento	-	Técnica de modelagem e implementação que separa os aspectos internos dos externos de um objeto. Esconder dados que representam detalhes da implementação.
E/S	-	Entrada e Saída.
TTL	-	Transistor-Transistor-Logic.
Buffer	-	Área de memória contínua utilizada para armazenar um dado.
Protoboard	-	Bancada física onde pode-se conectar componentes eletrônicos e efetuar suas ligações por conexão, provisóriamente.
Wire-wrapp	-	Variação de protoboard, onde as conexões são efetuadas por fios enrolados em terminais.

RESUMO

O uso de computadores para aprendizagem abre novas possibilidades no ensino de nível superior, possibilitando especialmente uma interação maior entre os alunos e as disciplinas de diferentes cursos.

O presente trabalho propõe e implementa um simulador adequado para o ensino de graduação de cursos de computação nas áreas de **hardware**, como circuitos digitais, arquitetura de computadores e automação industrial.

A análise, projeto e implementação foram baseados no paradigma da orientação a objetos, tomando como referência a teoria de sistemas. Esta abordagem possibilitou um desenvolvimento estruturado e resultou no sistema ***lab***GRAF de alta manutenibilidade e expandibilidade.

Uma análise de simuladores existentes mostra que os **software's** aplicativos destas áreas, utilizam ambientes específicos e incompatíveis entre si.

O simulador desenvolvido usa um ambiente único, onde é possível a simulação de sistemas híbridos compostos por dispositivos digitais e analógicos. Ao contrário de sistemas comerciais este simulador é aberto, permitindo a criação de componentes definidos pelo usuário e, adicionalmente proporciona a integração de equipamentos externos através de um interface de **hardware** e **software**. O próprio código é disponível, e pode servir como exemplo de programação de sistemas **C**, tal como, sistemas CAD e CAM.

ABSTRACT

The use of computers in the learning process opens new possibilities in the high level teaching, widening the interaction between the students and the disciplines of many different subjects.

This work proposes and implements a simulator which aids in the teaching of computer science undergraduation courses, particularly in the areas of hardware, digital circuits, computer architecture and industrial automation.

The analysis, design and implementation of the simulator were based on the object oriented paradigm, taking system theory as a basis. This approach enabled a structured development of a simulator, named labGRAF, which is both highly maintainable and extensible.

An analysis of existing simulator shows that application software in this area use both specific and mutually incompatible environments.

The implemented simulator uses an integrated workbench, which enables the simulation a hybrid systems composed of both analog and digital devices. This simulator is open, as opposed to commercial systems, thus it allows the addition of user defined components. It also permits the integration of external equipments by means of both a software and a hardware interface. The source code of the prototype is available for educational purposes and may serve as an example of **C** systems programming, such as **C**AD and **C**AM systems.

ÍNDICE

LISTA DE FIGURAS

1. INTRODUÇÃO

1.1. MOTIVAÇÃO

Através da rápida evolução tecnológica e a proporcional queda de preços de micro computadores, especialmente da linha IBM-PC®, estes têm encontrado grande aplicação nas mais diversas áreas comerciais e tecnológicas.

Evidencia-se, como exemplo, a área de Automação Industrial e suas subáreas Aquisição de Sinais, Comando e Controle. Estas áreas, bem como, o próprio equipamento utilizado, estão estreitamente ligados às disciplinas de Circuito Digitais, Arquitetura de Computadores e Automação Industrial (incluindo Controle de Processos), que fazem parte do currículo dos cursos de Computação de várias Universidades nacionais e internacionais.

Encontra-se diversas aplicações industriais utilizando PC em áreas tais como:

- Projeto (CAD - Computer Aided Design);
- Fabricação (CAM - Computer Aided Manufacturing e CNC - Controle Numérico por Computador);
- Controle de qualidade (CAQ - Computer Aided Quality Control);
- Integração (CIM - Computer Integrated Manufacturing);
- Treinamento (CAI - Computer Aided Instruction e CAT - Computer Aided Teaching);
- Planejamento de Produção (CAP - Computer Aided Planing).

O ensino exige a presença de disciplinas ligadas à estas aplicações, porém, existe uma mudança significativa no conteúdo de cada assunto, que tanto os alunos como os professores podem ter dificuldade no acompanhamento e aprendizado de novos conceitos.

Em geral, o entendimento dos conceitos ligados às áreas mencionadas, significam uma difícil barreira para estudante transpor durante o curso. Os alunos encontram

uma teoria complexa com aplicabilidade aparentemente limitada, tornando-os desmotivados no seu curso, como afirma Wellstead [59].

Uma solução para maximizar o aprendizado de uma matéria a ser ministrada e aumentar a eficiência do ensino, é a utilização de computadores. A eficiência aumenta considerando que o computador, por exemplo em comparação com livros ou aulas expositivas, possibilita uma interação maior e uma adaptação aos conhecimentos do aluno, obtendo-se assim um aprendizado mais individual.

Também para minimizar as dificuldades encontradas durante o curso, pode-se ministrar aulas práticas apoiadas por computador.

Existem várias formas de utilização ou aplicação do computador no ensino, dentre as quais destaca-se o aprendizado através de: fatos; descobertas e resolução de problemas.

O aprendizado através de *fatos* divide-se em duas partes: a primeira é feita por tutoriais, onde o sistema determina o caminho a seguir; e na segunda, efetivada por sistemas de treinamento, o aluno determina o caminho a ser tomado pelo sistema computacional.

A aprendizagem por *descoberta* é possibilitada por sistemas mais complexos, como sistemas de simulação com animação gráfica. Neste tipo de aprendizagem, utiliza-se também, jogos com a filosofia da simulação.

Complementando a fase de aprendizagem, existem os sistemas que ensinam através da *resolução de problemas*.

A tendência dos alunos gostarem de experimentar conceitos, especialmente apoiados por sistemas de simulação ou jogos, e ainda mais, considerando que simuladores tradicionalmente são usados nas áreas técnicas, resulta na principal tarefa deste trabalho, que é desenvolver um ambiente de simulação, a ser utilizado por alunos dos cursos da área

tecnológica. O sistema será apresentado de forma *aberta*, proporcionando facilidades para sua ampliação e também para o desenvolvimento de novas ferramentas.

1.2. A UTILIZAÇÃO DE SIMULADORES NO ENSINO SUPERIOR NA ÁREA DE COMPUTAÇÃO

De forma simplificada, as disciplinas dadas em cursos de computação podem ser classificadas em duas grandes áreas:

- área de **hardware;**
- área de **software**.

Enquanto disciplinas da área de **hardware** tratam do funcionamento de computadores, as de **software**, estão ligadas à sua programação.

Aulas de **hardware**, tradicionalmente estão contidas nas disciplinas de Circuitos Digitais e Arquitetura de Computadores e com a crescente importância da área de Automação Industrial, ela começa a fazer parte dos cursos de Bacharelado em Ciência da Computação e Engenharia da Computação. Já, disciplinas como Sistemas Operacionais e Computação Paralela, possuem uma forte interligação entre **software** e **hardware**.

Disciplinas da área de Automação Industrial, podem ser orientadas mais diretamente para o **hardware**, enfocando o conteúdo de controle de dispositivos externos; ou mais para o **software** voltado para os sistemas C (CAD, CAM, etc.).

Este trabalho tem aplicação às disciplinas ligadas ao **hardware**, por se tratar de uma área carente de novos recursos para o ensino. Um forte motivo para tal, é o fato dos alunos de computação, freqüentemente mostrarem dificuldades no aprendizado de circuitos eletrônicos envolvendo dispositivos digitais e analógicos, e aplicados a arquitetura de computadores e automação industrial.

As aulas de Circuitos Digitais, Arquitetura de Computadores e Automação Industrial, abordam em geral itens como:

- Circuitos Digitais:
  - Álgebra de Boole;
  - Circuitos Combinacionais;
  - Circuitos Seqüenciais;
- Arquitetura de Computadores:
  - Arquitetura interna da CPU;
  - Conjunto de instruções;
  - Barramentos;
  - Memória;
  - Comunicação com o mundo externo (interfaces).
- Automação Industrial:
  - Medição (conversores A/D e D/A);
  - Comando (sistemas discretos ou digitais);
  - Controle (sistemas contínuos ou analógicos e discretos);
  - Sistemas C (CAD, CAM, CIM, etc.).

Em geral estas disciplinas são ministradas, dividindo-as em aulas teóricas (ou expositivas) e práticas (ou de laboratório).

Nas aulas expositivas aborda-se a teoria relativa aos temas específicos, através da utilização do quadro negro e retro projetor.

Nas aulas práticas, são implementadas experiências freqüentemente baseadas em componentes eletrônicos, mantidos a disposição dos alunos, que fazem suas devidas conexões utilizando **protoboard**, **wire-wrapp** ou algum outro sistema onde se possa conectar os componentes, e efetuar suas ligações de forma provisória.

Servindo de entrada para estes circuitos, são utilizados fontes de sinais como chaves liga/desliga ou geradores de funções, e para visualização dos sinais, utiliza-se dispositivos como **led's**, multímetros, frequencímetros, osciloscópios e analisadores lógicos.

Este método, utilizado atualmente para aulas práticas, apresenta alguns aspectos *pedagógicos* e *econômicos* que acredita-se, serem contraproducentes para o ensino.

Na montagem de circuitos com componentes eletrônicos, o aluno depara-se muitas vezes com um emaranhado de fios tão complexo, que facilmente perde a visão do que está fazendo, e freqüentemente, perde muito tempo procurando erros, correndo constantemente o risco de queimar componentes. Estes fatores afastam-no do objetivo central que é o aprendizado da experiência.

Na simulação, o circuito apresenta-se mais claro, dado que é construído e testado no monitor de vídeo de um computador. Esta característica evita a perda de tempo com a montagem do circuito, permitindo uma maior concentração no desenvolvimento e teste do projeto eletrônico.

Muitas vezes, pelo fato de um ou mais alunos perderem a aula prática, o professor se vê obrigado a reorganizar toda a experiência, para que esta possa ser aplicada aos alunos faltosos. Já, com a utilização de um sistema de simulação, o programa estaria sempre a disposição no computador, que poderia estar até mesmo na casa do aluno.

Resumindo, este estudo evidencia dois fatores favoráveis à simulação:

- a ausência de perigo em destruir algo;
- e a disponibilidade constante do sistema de simulação.

Este fatores, contrastam com as características negativas, quando se usa a prática convencional baseada em **protoboard** ou semelhante, e podem ser aproveitados para reorganizar a estrutura atualmente utilizada para ministrar aulas.

Baseando-se no interesse dos alunos por atividades experimentais, é possível conseguir-se um processo de aprendizagem mais sólido, dividido em três fases como segue:

- a primeira fase envolve *experimentação* com dispositivos eletrônicos e demais componentes;
- a segunda, a explicação da *teoria* utilizando modelos construídos pelo professor, considerando as experiências da primeira fase;
- a terceira, trata-se da *aplicação* dos conhecimentos adquiridos para resolver problemas.

Quanto ao aspecto econômico, evidencia-se a necessidade do alto investimento em componentes eletrônicos, e equipamentos para geração e medição de sinais, já que deve-se manter uma bancada de experimentos para cada equipe, que por sua vez, é formada com o menor número de alunos possível.

Os investimentos podem ser significativamente reduzidos com a utilização de programas de simulação, onde os sistemas físicos são implementados e analisados virtualmente através do micro computador, imaginando-se ter um estoque de componentes ilimitado e uma variedade de equipamentos de medição.

Atualmente o micro computador, comparado por exemplo a um osciloscópio, tem preço baixo, e é um equipamento que está disponível na maioria dos cursos, não só de computação, mas em toda Universidade.

Num simulador então, o aluno implementa o circuito e interliga os componentes na tela do computador (programação visual) e a seguir, observa o comportamento de saída do circuito.

A simulação torna-se mais interessante, incluindo-se no próprio ambiente, a possibilidade de controle de sistemas externos em tempo real, através de uma interface física apropriada, como mostra a figura 1.

Nesta figura, observa-se um diagrama de blocos representando a conexão de um circuito virtual (1) com um circuito físico (5) através de uma interface de **hardware** (3), a qual é ligada ao computador via expansor de barramento (2) e ao circuito real, através de portas de comunicação (4).

Em uma configuração deste tipo, onde o sistema de simulação e a interface operam conjuntamente como um *instrumento de automação*, podem ser realizados experimentos reais como medir a temperatura de um forno ou a velocidade de um motor, e atuar sobre os mesmos mantendo-os em seus pontos de operação previamente estabelecido pelo operador.

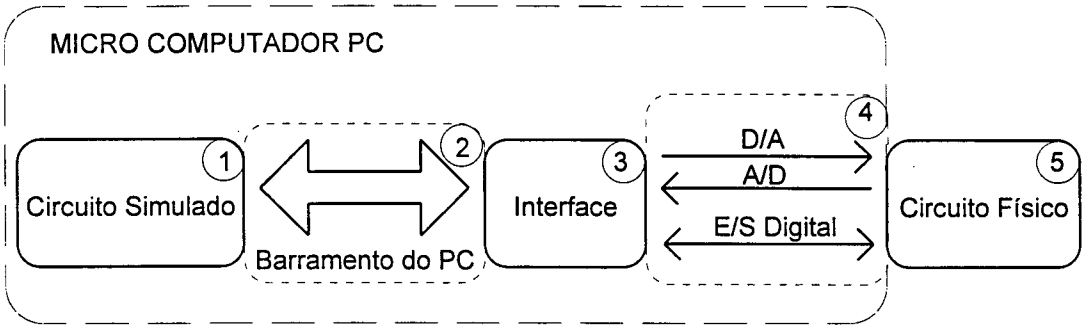


Figura 1. Diagrama de blocos da conexão do micro com circuitos físicos.

1.3. COMPONENTES DE UM SIMULADOR

Um simulador, segundo Bodendorf [5], pode ser dividido em seis partes de acordo com as tarefas executadas. Como mostra a figura 2 estas partes são:

- INTRODUÇÃO (1), nesta fase é determinado o objetivo do sistema e a forma ou formas de sua utilização, isto em geral, é feito através de uma ferramenta de auxílio ao usuário (**help**) ou mais detalhadamente, através do manual do usuário;
- CENÁRIO (2), onde o usuário pode compor graficamente o circuito e parametrizar os dispositivos;
- INICIALIZAÇÃO DA AÇÃO (3), nesta fase a simulação é inicializada através do teclado, **mouse** ou **joystick**.

- AÇÃO (4), é a fase da simulação, permitindo a intervenção pelo usuário, podendo-se determinar a velocidade da evolução da simulação e alterar a parametrização dos dispositivos simulados;
- REAÇÃO (5), onde os resultados da simulação são apresentados;
- FINALIZAÇÃO (6), onde pode-se encontrar métodos para finalização da simulação; para salvamento do circuito e para saída do sistema.

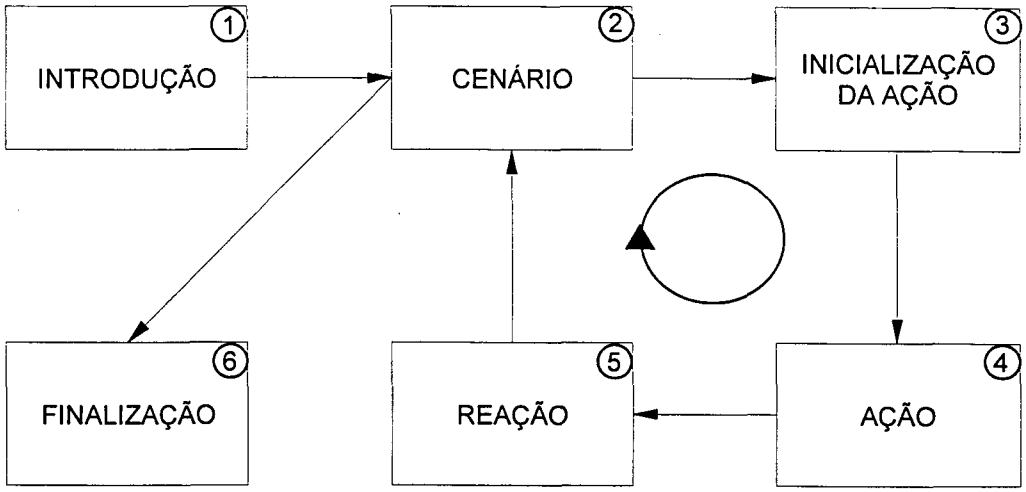


Figura 2. Componentes de um simulador segundo Bodendorf.

1.4. DIRETRIZES DE TRABALHO

Pretende-se neste trabalho, explorar as vantagens que os simuladores proporcionam para o ensino na área de **hardware**/automação industrial em cursos de computação.

O objetivo é propor e implementar, um simulador que permita simulações nas áreas de Circuitos Digitais, Arquitetura de Computadores e Automação Industrial, em um único ambiente de trabalho.

O programa de computador que implementará o simulador, deverá ser aberto, permitindo aos alunos efetuarem mudanças em seu código fonte, servindo assim, como modelo para o aprendizado de sistemas C (CAD, CAM, etc.).

O trabalho baseia-se na análise de simuladores disponíveis no mercado nacional e internacional, apresentada no próximo capítulo que é finalizado com uma proposta de trabalho mais detalhada (item 2.5).

2. SIMULADORES PARA SISTEMAS DISCRETOS, ARQUITETURA DE COMPUTADORES E SISTEMAS CONTÍNUOS

Como justificado a seguir, denomina-se de forma mais geral, circuitos digitais como sistemas discretos, englobando neste tipo de sistema os computadores digitais. Assume-se que na automação industrial, são utilizados tanto os sistemas discretos como analógicos, que são generalizados aqui como sistemas contínuos. Para estes dois tipos de sistemas existem no mercado simuladores que podem ser classificados como:

- educacional;
- semi-profissional;
- profissional.

Neste capítulo são estudados alguns exemplos característicos, diferenciando-os conforme sua aplicação nas áreas de Sistemas Digitais, Arquitetura de Computadores e Automação Industrial, enfatizando-se os sistemas aplicados na educação.

2.1. SIMULADORES DE SISTEMAS DISCRETOS

Os simuladores de circuitos digitais podem ser divididos em:

- simuladores que se baseiam na descrição de circuitos por álgebra booleana ou em uma linguagem de simulação[1];
- simuladores que utilizam interfaces gráficas, que permitem implementar e apresentar o circuito no monitor de vídeo do computador.

a) LÓGICO [6]

Implementado na FEE-UNICAMP por Ivanil Bonatti e Marcos Madureira.

Define-se este simulador como predominantemente educacional, baseando-se no fato de compor parte de um livro [6], e sugerido para aplicação no ensino.

---

[1] Linguagem de simulação é uma linguagem de programação voltada para descrever sistemas físicos de forma virtual.

Este simulador trabalha com uma linguagem de simulação, com comandos que simulam portas lógicas básicas, como a da figura 3, e estruturas mais complexas de circuitos combinacionais e seqüenciais.

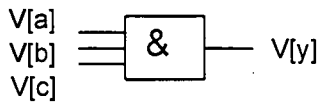


Figura 3. Função **E** de 3 entradas.

Os elementos são representados pela estrutura de dados **V** (figura 3 e listagem 1) que é uma cadeia de valores, que contém os valores das entradas e saídas das portas em um dado instante, e um algoritmo que realiza uma função lógica, como o ilustrado na listagem 1.

```
Procedure NAND3(a,b,c,y)
begin
      E := V[a]*V[b]*V[c];
      if E>=2 then E:=2 else E:=1-E;
      (*Coloca V[y]:=E na fila*)
end;
```

Listagem 1. Rotina que implementa a função **NÃO E (NAND)** de três entradas.

Os autores utilizaram três valores (0, 1 e 2) para representar respectivamente os níveis lógicos baixo, alto e indeterminado. Todos os dispositivos simulados são inicializados com o valor indeterminado.

Por exemplo, suponhamos na porta lógica **E**, ilustrada na figura 3, e implementada no algoritmo da listagem 1, que V[a]=1, V[b]=1 e V[c]=2 (indeterminado); V[y] recebe o valor 2, pois a variável auxiliar 'E' (listagem 1), após a multiplicação, resultou em 2 ou indeterminado. De acordo com este algoritmo, a saída é 0 se pelo menos uma entrada for 0, e 1 quando todas forem 1.

O fato de existir resultados indeterminados no cálculo do valor de saída, gera uma instabilidade no circuito. Os cálculos são feitos repetidamente, até que o circuito se estabilize ou esgote o tempo de simulação programado pelo usuário (figura 4 parte 3). A

estabilização de um circuito, significa a obtenção de valores nas saídas dos componentes, iguais ao do ciclo de relógio anterior. Estes resultados podem ser zero ou um, porém necessariamente diferentes de indeterminado.

Para a montagem do circuito virtual, que é composto por vários elementos interligados, o usuário utiliza *comandos* como os das figura 4 parte 2. Estes comandos passam parâmetros para o vetor **V**, utilizado em rotinas semelhantes à listagem 1. Uma seqüência destes comandos devem estar gravados em um arquivo texto com extensão ".CIR", do qual o LÓGICO lê as informações para efetuar a simulação.

O LÓGICO convenciona o sinal dos índices do vetor como negativos para entradas externas e positivos para os estados internos, ou seja, valores gerados pelo circuito (veja exemplo na figura 4 parte 1 e 2). Em geral, o dado presente na entrada de uma porta, quando proveniente da saída de outro componente, é representado por um parâmetro positivo.

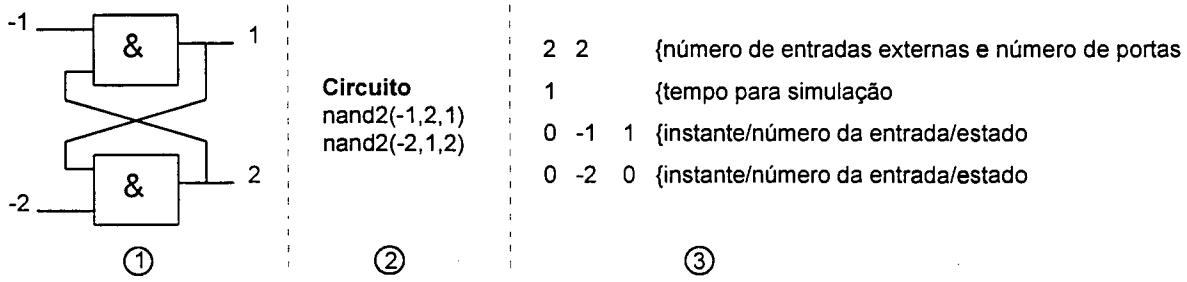


Figura 4. Estrutura de uma célula básica de memória tipo set/reset (1) sintaxe da linguagem para arquivo ".CIR" (2) e dos valores para as entradas externas no arquivo ".OND" (3).

Todos os dados que representam entradas externas ao circuito, são fixados em um arquivo com extensão ".OND". O resultado da simulação é apresentado na forma de tabela verdade, ou na forma de diagrama de tempo.

b) LOGICLAB [3]

Proposto em um artigo da revista Dr. Dobb's Journal, é um simulador lógico de nível educacional, visto a publicação do código fonte, implementado na linguagem orientada a objetos SMALLTALK.

A simulação pelo LogicLab é feita em um ambiente gráfico e dividida em duas fases:

- na primeira, o usuário instala os componentes e efetua sua ligações;
- na segunda, é feita a simulação do circuito;

Os componentes são implementados em classes nas quais são definidas as entradas, as saídas e as funções. A publicação não fornece maiores detalhes sobre a forma de implementação dos componentes.

As *interligações* que representam as conexões elétricas dentro do circuito são representadas por uma classe chamada *nodos lógicos*.

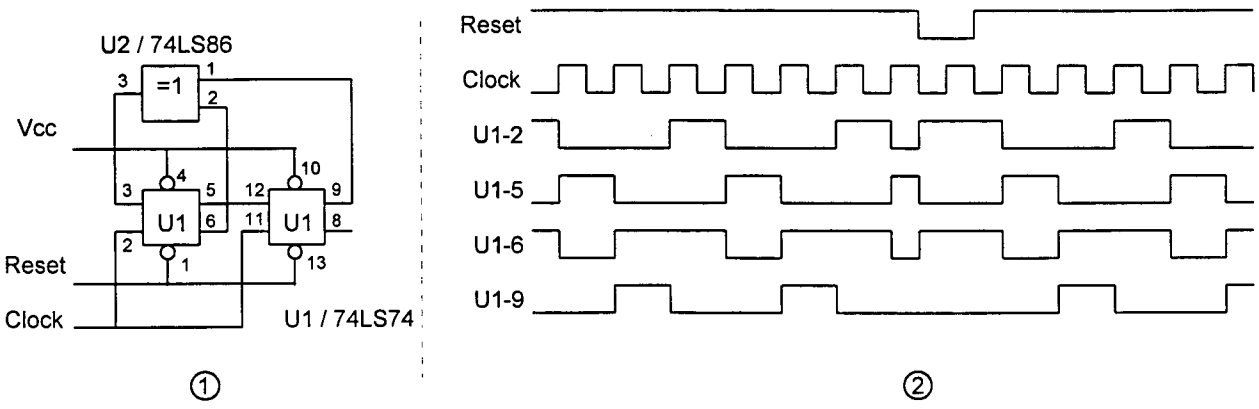


Figura 5. Simulação de um circuito contador divisor por três: diagrama esquemático (1) e diagrama de tempo (2).

Para alimentar as entradas, existem geradores de sinais e para verificar a saída existem pontas de provas, ambos virtuais. Os últimos podem ser conectados em qualquer ponto do circuito (figura 5 parte 1) e mostram resultados graficamente em um diagrama de tempo (figura 5 parte 2).

Faz parte da publicação, uma listagem do código fonte do programa. Este foi implementado em um ambiente SmallTalk, porém, durante a sua execução, após a abertura de uma janela (supostamente a principal) com o ambiente de trabalho, constatou-se que não era

possível instalar qualquer dispositivo, montar um circuito, ou mesmo visualizar-se as pontas de provas. Concluiu-se então, que a listagem fornecida estava incompleta, desta forma, tanto a análise como a reprodução da figura 5, são feitas segundo as explicações do artigo escrito.

c) LOGSIM [25]

Este produto oferecido pela empresa Alemã **Graf Elektronik Systeme** é proposto para o nível educacional.

A composição do circuito é feito, através de um ambiente gráfico, onde existem menus que mostram uma biblioteca de componentes. O usuário pode selecionar os itens dos menus com utilização do mouse e fixá-los na posição desejada dentro do ambiente de trabalho .

As conexões são feitas também com a utilização do **mouse**, através do qual marca-se o primeiro elemento pressionando um dos botões, e o usuário o movimenta até a posição desejada pressionando uma segunda vez o botão para finalizar a operação.

Durante o movimento do **mouse**, a linha que representa a conexão, é mostrada acompanhando o seu movimento.

Como elementos de entrada e de saída, existem respectivamente, chaves e **led's** virtuais, osciladores e osciloscópio, que permitem uma apresentação semelhante ao do LogicLab.

d) ORCAD/VST [45]

Oferecido pela empresa OrCAD **System Corporation**, faz parte de um conjunto de "ferramentas" para desenvolvimento e análise de placas de circuito impresso a nível profissional semi-profissional.

O OrCad/VST (Verification and Simulation Tools) tem uma linguagem própria para modelagem de sistemas digitais, que se preocupa com os componentes e suas ligações, considerando também tempos de atraso.

Sinais de entrada são definidos com um editor chamado Stimulus, e o resultado da simulação é apresentado na forma de um oscilograma.

O OrCad/VST traz grandes vantagens no desenvolvimento de sistemas quando ele é usado junto com o OrCad/SDT (Schematic Design Tools), que é uma ferramenta gráfica para projetos de Sistemas Digitais. Com o OrCad/SDT pode ser gerado um arquivo chamado **net-list** contendo informações sobre os componentes e ligações do circuito. Este arquivo pode ser usado diretamente pelo OrCad/VST, substituindo a definição do circuito pela linguagem de modelagem.

2.2. SIMULADORES DISCRETOS DE ARQUITETURAS DE COMPUTADORES

Simuladores de arquiteturas de computadores podem são classificados em dois grupos:

- os que usam modelos projetados especificamente para o ensino;
- e os que usam uma máquina comercialmente estabelecida.

Com o computador projetado para o propósito de ensino, pode-se construir uma arquitetura com um conjunto de instruções reduzido e extremamente regular, como as propostas por Gibson [23], Jones [28], Malvino [36] e Taub [57], que são arquiteturas simples, a nível didático.

Numa complexidade crescente encontra-se, simuladores e depuradores (**debbugers**) comerciais, indicados para estudantes com nível mais avançado, mas nunca são sugeridos na literatura como úteis para a fase inicial de um curso, como mencionado por Silvester [53].

a) ECE [21]

Apresentado em um artigo da IEEE Transactions on Education, o ECE (Educational Computer Emulator) é aplicado no curso **Introduction to Computer Engineering** da Universidade de **Illinois**, e direcionado ao estudo de arquitetura de computadores.

O ECE é um emulador com animação gráfica baseado em estações de trabalho com monitores de vídeo de alta resolução (**workstation's**). Foi implementado no **AT&T UNIX PC's**, bem como no **DEC** e no **IBM workstation** sob o **X-Windows.**

Este simulador consiste então, de uma CPU hipotética, cuja arquitetura interna é formada por um acumulador de 21 bits, sete registradores de uso geral (R1 a R7) de 12 bits, registrador de instruções (IR), contador de programa (PC) e uma ALU com 16 funções e **flag** de **overflow**.

O formato da linguagem **assembly** é fixo, onde cada instrução tem um código de 21 bits de tamanho.

A CPU do ECE tem uma arquitetura que acessa 4K de 21 bits na memória RAM simulada, conectada ao processador por um barramento de dados do mesmo tamanho e por um barramento de endereços de 12 bits, além disso, usa uma ROM com 512 palavras de 32 bits cada, para armazenamento do micro-código das instruções.

Além do ECE aceitar que o usuário escreva um programa na sua linguagem **assembly**, que será montada diretamente na memória RAM simulada, existe a possibilidade de emular o seu micro código já existente na ROM, bem como possibilidades de escrever e testar um novo micro código, análogas às oferecidas para os programas em linguagem **assembly**, criando novas instruções.

Existem 28 posições da ROM, ocupadas por micro códigos, e os restantes dos seus 512x32 bits estão disponíveis para o usuário escrever seu próprio micro código. As micro

sequências armazenadas na ROM podem ser mostradas na janela da memória e executada individualmente.

Todos os componentes da CPU emulada são mostrados na tela gráfica, onde são visualizados os valores contidos em cada componente e sua ligações através de barramentos representados por linhas genéricas, sem especificação da quantidades de fios que passam por ela. Além disto é apresentada uma janela com 17 posições consecutivas da memória RAM ou ROM (exclusivamente).

b) COMPUTER-AIDED TEACHING PACKAGE [16]

Apresentado em um artigo da **IEEE Transactions on Education**, é um pacote de programas educacionais, desenvolvido em Turbo Basic, que simula a Unidade Central de Processamento Z80, disponível comercialmente.

Ele possui quatro modos de operação: (1) demonstração; (2) aprendizado do conjunto de instruções; (3) entrada de um programa em linguagem **assembly** ou (4) código de máquina, para execução.

O usuário tem acesso aos quatro modos de operação, na seguinte forma:

No modo de *demonstração* (1) é feita a apresentação de como o usuário pode usar o conjunto de instruções, entrar e executar um programa

No modo para *aprendizado* (2) do conjunto de instruções da CPU Z80, o usuário tem acesso aos mnemônicos de todo o conjunto de instruções, apresentados em ordem alfabética no monitor de vídeo.

Selecionando uma instrução, são apresentados as informações relevantes sobre a mesma. Pode-se também executar a instrução selecionada, neste caso a tela mostra a representação gráfica da CPU e evidencia suas transições de estado do ciclo de execução.

O usuário também pode dar *entrada a um programa* em **assembly** (3) ou em linguagem de máquina (4) completo e executá-lo. Neste caso, deve-se escrever o programa em um editor de textos, carregá-lo no sistema, fazer a compilação e em seguida executá-lo passo a passo.

Durante a inicialização do **software** é desenhada a representação gráfica do Z80 numa tela gráfica com a largura de 80 colunas do tipo texto, representando tanto a arquitetura da CPU, com a ULA e outros componentes internos, bem como as conexões internas .

Alternativamente, através de um rolamento horizontal da tela, as portas de E/S e a memória principal podem ser visualizadas, com as devidas conexões através de barramentos.

Para simulação dos sinais de barramento, são utilizados códigos de cores, onde as cores dependem, dos dados presentes em cada linha do barramento (vermelho = 1, verde = 0 e branco = indeterminado). O sentido do movimento dos dados é indicado por pequenos traços, que se deslocam sobre as linhas do barramento ativo, desde a origem até o destino.

Os registradores são simulados, de forma a visualizar o movimento de dados entre eles da seguinte forma: o registrador fonte pisca primeiro, e em seguida o registrador destino reage da mesma forma, e o barramento de dados indica o sentido da transferência como descrito no parágrafo anterior.

A simulação da memória é dividida em três seções, apresentadas na tela: uma área de programa com sete posições sucessivas; uma área de dados com três posições e uma área de pilha com os cinco endereços mais ao topo desta.

Para as portas de E/S, quando é executado uma instrução deste tipo, é mostrado o endereço da porta acessada e o dado a ser transferido (no caso de uma instrução de entrada é solicitado ao usuário que este forneça o dado, simulando uma aquisição).

No canto esquerdo inferior da tela, aparece o diagrama de tempo de alguns sinais de controle traçados a cada ciclo de relógio (**clock**) durante a execução da instrução, são eles:

- **CLK (Clock)**, indicando os pulsos do relógio que sincroniza os componentes;
- **MREQ (Memory Request)**, que indica um endereço válido no barramento para leitura ou escrita na memória;
- **IORQ (I/O Request)**, indica um sinal de controle para efetuar uma transferência de dados da/para CPU para/de um dispositivo externo;
- **RD (Read)**, indica que a CPU quer ler dados da memória ou de dispositivos de E/S;
- **WR (Write)**, indica o barramento de dados tem um dado válido para ser escrito na memória, ou dispositivos de E/S;
- **M1 (Machine cicle one)**, junto com MREQ, indica que o ciclo de máquina corrente é de busca de instrução (opcode fetch) e junto o IORQ, indica um ciclo de reconhecimento de interrupção;
- **RFSH (Refresh)**, junto com MREQ, indica que o set bits mais baixos do barramento de endereços, podem ser utilizados para refrescar a memória dinâmica.

2.3. SIMULADORES PARA AUTOMAÇÃO INDUSTRIAL

Simuladores para Automação Industrial podem ser classificados em dois tipos:

- simuladores para resolução de *sistemas discretos*;
- simuladores para resolução de *sistemas contínuos*.

2.3.1. SISTEMAS DISCRETOS

O dispositivo discreto mais popular (e de importância destacada) da área de automação, é o Controlador Lógico Programável (CLP), que é um computador especializado para execução de funções booleanas, utilizado para controlar processos combinacionais e seqüenciais, dependentes de estados.

Para programar os CLP's, existem linguagens baseadas em comandos escritos em inglês, porém, o suporte mais utilizado é a linguagem gráfica (**Ladder Diagram**) com a

capacidade de representar cada operação booleana que o CLP pode executar, bem como sua seqüencialidade.

Em geral as empresas fornecedoras, oferecem programas de treinamento que se baseiam na simulação do controlador estudado. Estes simuladores, em sua grande maioria, são tradutores da linguagem gráfica ou outra de alto nível, para a linguagem **assembly** utilizada pelo CLP, e a simulação é feita baseada na visualização do conteúdo dos seus registradores.

Na área de planejamento de produção (CAP - Computer Aided Planing), são utilizados simuladores clássicos, que se baseiam em eventos. Eles simulam seqüências de processos que executam tarefas, e são regidos por eventos, considerando a disputa de recursos. As tarefas dos processos podem ser bloqueados em filas, onde aguardam até que os recursos sejam liberados. Estas filas podem ser ordenadas através de vários métodos como LIFO, FIFO ou de prioridade definida especificamente para cada entidade. A mudança de estado, ocorre instantaneamente em intervalos de tempo descontínuos e geralmente imprevisíveis. Os tempos entre os eventos, são determinados estatisticamente, de acordo com algum número randômico, ou seja, aleatóreamente [24].

O SIMSCRIPT [9] e o SIMAN [46], são exemplos de simuladores discretos voltados para a área Automação Industrial que utilizam esta técnica.

2.3.2. SISTEMAS CONTÍNUOS

a) FLOWLEARN [17]

Este produto, oferecido pela empresa Alemã E+PK Ingenieurbüro, se propõe a atender projetos de sistemas contínuos realizados através de uma interface gráfica com o usuário.

Neste ambiente, existe uma tela que é dividida matricialmente, formando posições fixas, onde o usuário pode instalar, com apoio do teclado, apenas uma caixa por

posição. Estas caixas instaladas visualmente, representam os módulos do sistema a ser desenvolvido.

Após a instalação da caixa, passa-se à definir as funções dos módulos. Estas são definidas através de um menu, que fornece a possibilidade do usuário escolher em uma biblioteca, várias funções analógicas como integradores, somadores, derivadores, etc. e um pequeno conjunto de funções digitais utilizadas para comando de elementos do sistema contínuo projetado. Todos os módulos podem ser parametrizados.

Sobre a moldura das caixas, existem pontos visuais utilizados para efetuar a conexão entre os objetos. Esta conexão é feita pela movimentação do cursor via teclado; é necessário marcar primeiramente o elemento que receberá o dado em sua entrada e depois a saída do elemento que fornece um resultado parcial. Feito isto a linha de conexão é fixada.

As entradas são simuladas por elementos virtuais como chaves, geradores de pulso, geradores de frequência e outros selecionados numa biblioteca.

O resultado é apresentado em elementos simulados como o osciloscópio que mostra um gráfico animado, ou de forma auxiliar em lâmpadas ou **led's e alto-falantes.**

O fabricante oferece de forma opcional, e sob encomenda, a possibilidade de interfaceamento com o ambiente externo, porém, com custos altíssimos.

b) LABTECH [32]

Fornecido pela Labtech, uma empresa Americana, é um conjunto de programas orientados à automação de ensaios de laboratório.

O sistema trabalha com uma interface gráfica, através da qual, o usuário escolhe elementos virtuais em uma biblioteca. Estes elementos são representados por caixas previamente associadas às suas funções, e são acessados através de ícones e menus, com a

utilização do **mouse** ou teclado. Para compor o sistema, o usuário seleciona no menu, o componente e instala-o em qualquer lugar. Não existem posições pré definidas.

As funções dos elementos virtuais podem ser parametrizadas.

A conexão entre os elementos é efetuada com o posicionamento do **mouse** sobre os elementos e os respectivos **clicks**, o que provoca o desenho da linha que representa estas conexões. O algoritmo utilizado para traçar estas linhas, não permite que o usuário participe da definição do caminho a ser tomado pela linha, o que na maioria das vezes causa uma sobreposição desordenada destas linhas, provocando uma poluição visual indesejada.

Os resultados são apresentados numa tela gráfica exclusiva, ou em alguns casos, em janelas montadas junto ao circuito obtido. Os gráficos são apresentados em forma de barras ou linhas, dependendo da natureza da operação.

O Labtech admite também que seja feita aquisição de dados através de uma interface de **hardware** adquirida, porém, o usuário é obrigado a encomendar o **software** específico para controlar esta interface. Além disso, a empresa atende a encomendas sobre um grupo reduzido de fornecedores de placas.

De forma opcional a empresa oferece um software adicional para monitoração e controle de manufatura. Sua interface gráfica permite a utilização de símbolos que representam objetos físicos, por exemplo, válvulas, tanques, misturadores, etc.

Após o circuito montado de forma similar ao circuito que utiliza caixas funcionais, obtém-se uma figura, por onde é possível acompanhar a evolução do processo com a animação destes símbolos, demonstrando os movimentos dos componentes físicos.

2.4. CONSIDERAÇÕES SOBRE OS SISTEMAS SIMULADORES EXISTENTES

Após uma análise dos simuladores apresentados anteriormente, pode-se concluir que:

a) existem no mercado simuladores para as três áreas: circuitos digitais e arquitetura de computadores (sistemas discretos) e automação industrial (discretos e contínuos), porém, é exigido para cada área um ambiente próprio e exclusivo. Somente os sistemas FlowLearn e LabTech, que são projetados originalmente para simulação contínua, possuem algumas funções discretas de apoio;

b) a maioria dos sistemas analisados possuem uma biblioteca de dispositivos limitada, ou seja, o usuário não tem a possibilidade de criar seus próprios elementos;

c) alguns sistemas como o FlowLearn e o LabTech, podem integrar o ambiente através de um interface de **hardware**, porém, com custos muito elevados.

d) existem diferenças na montagem do *cenário*, e para tal, podem ser utilizados uma linguagem de programação específica ou interface gráfica. As interfaces gráficas distinguem-se em dois tipos: no primeiro, escolhe-se dentro de um conjunto de ícones fornecido pela interface, o elemento necessário, e o posiciona no lugar desejado; no segundo, posiciona-se uma caixa vazia no lugar adequado, e após isto, determina-se as funções e parâmetros. Uma última variação encontrada nas interfaces, é o fato de algumas permitirem instalar os componentes em qualquer lugar, já outras apenas em lugares pré-fixados, utilizando exclusivamente o teclado ou o teclado em conjunto com o **mouse**;

e) a maioria dos sistemas são comerciais, desta forma, não se conhece o código fonte, porém, pode-se observar pelas características do FlowLearn, e pela parte do código do LogicLab que foram programados por uma linguagem orientada à objetos.

2.5. PROPOSTA DE TRABALHO

No presente trabalho, propõe-se um ambiente de simulação, que visando usar os pontos positivos dos sistemas pesquisados, amplie as possibilidades dos simuladores existentes, produzindo as seguintes principais características:

a) O sistema proposto será *homogêneo*, no sentido de simular sistemas discretos (desde circuitos booleanos até CPU's) e contínuos, excluindo-se os sistemas de simulação discreta (CAP), em um só ambiente. Esta característica, representa a integração entre os dois tipos básicos de modelos de sistemas físicos. Busca-se com isto, suprir a necessidade desta integração para facilitar a aprendizagem;

b) Através deste sistema, o usuário poderá criar novos componentes, ampliando assim, a biblioteca fornecida;

c) O sistema proposto envolverá a simulação e o interfaceamento com o circuitos reais em um só ambiente. Terá uma interface de hardware totalmente documentada, permitindo ao usuário um tratamento mais interessante dos sistemas simulados, com possibilidades de aplicação em sistemas automáticos reais. Além disso, o aluno ou outro usuário, poderá construir suas próprias interfaces de **hardware** para comunicação do simulador com o ambiente externo, o que é considerado uma experiência muito interessante, porque computadores são utilizados com frequência na automação;

d) O sistema proposto fornecerá ao usuário, facilidades de utilização através de *interface gráfica* apoiada por menus, que flutuarão sobre posições pré-fixadas do ambiente de trabalho, onde serão instalados os componentes;

e) Fará parte do pacote que envolve o sistema proposto, o fornecimento de *documentação* suficiente para que o usuário possa efetuar pesquisas sobre este, modificando as rotinas existentes e implementando novas, fato interessante para os alunos principalmente de cursos de computação, porque lhes permite o aprendizado sobre as estruturas de programação existentes, principalmente em CAD's, dado que o sistema será desenvolvido em uma linguagem orientada a objetos.

Este projeto, especialmente o propósito de integrar sistemas discretos e contínuos em um só ambiente, parece ambicioso, porém, durante o decorrer do trabalho será mostrado que

com aplicação de métodos orientados a objetos e com especificações sistemáticas, efetuadas a partir da teoria de sistemas, pode-se chegar à uma conclusão vitoriosa. O trabalho será facilitado pela adoção de conceitos de interface com o usuário, retirados dos pacotes existentes, procurando interfaces simples e práticas como a utilizada pelo FlowLearn.

Os tarefas desenvolvidas são descritos a seguir, nos próximos 4 capítulos. No capítulo 3, é apresentado o Paradigma da Orientação a Objetos (POO), que forma a base para os capítulos 4, 5 e 6.

O capítulo 4, primeiramente apresenta a análise técnica, com base na teoria de sistemas, dos sistemas físicos relacionados com o problema, e numa segunda fase, baseados no resultado da fase anterior, identifica os objetos relacionando suas características comuns, obtendo assim, uma identificação das suas classes.

O capítulo 5 descreve o projeto detalhado das classes, e o capítulo 6 os aspectos interessantes sobre a implementação do simulador ***lab*GRAF**, que é o sistema computacional resultante deste trabalho.

O capítulo 7 apresenta finalmente a análise dos resultados e o capítulo 8 a conclusão do trabalho.

3. O PARADIGMA DA ORIENTAÇÃO A OBJETOS

Orientação a objetos é uma forma recente de concepção de software que utiliza como elemento central o objeto.

Esta concepção, em contraste com as técnicas convencionais, que decompõem o problema em estruturas de dados e procedimentos isolados (figura 6 parte 1), combina estruturas de dados e comportamento em uma única entidade (figura 6 parte 2), e além disso, unifica o tratamento dos elementos do problema, em relação aos seus dados e procedimentos que atuam sobre ele.

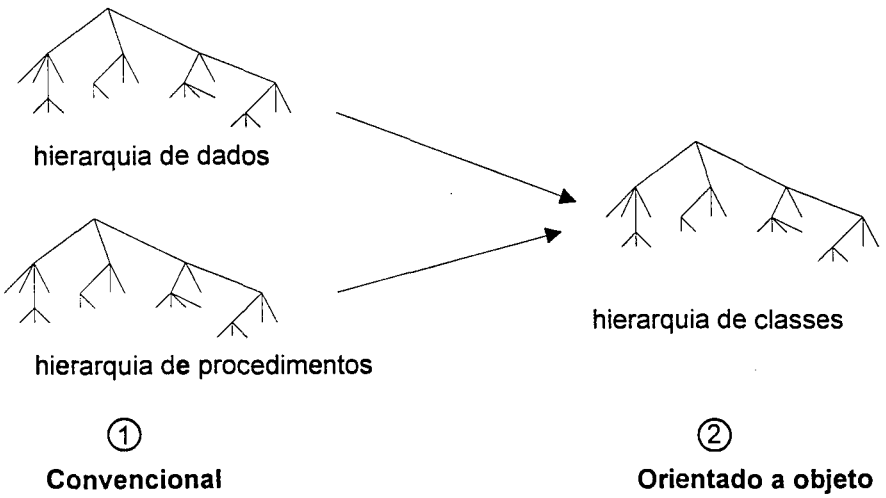


Figura 6. Comparação das hierarquias

De forma simplificada, a orientação a objeto se preocupa com quatro aspectos fundamentais: objeto, classificação, herança e polimorfismo, de acordo com Rumbaugh [50].

3.1. OBJETOS

Visto pela perspectiva da cognição humana, segundo Booch [7], um objeto é uma entidade com uma das seguintes características:

- algo visível e/ou tangível;

- algo que possa ser capturado pelo intelecto;
- algo a que uma ação possa ser dirigida ou pensada.

Na construção dos sistemas computacionais, os objetos de interesse são aqueles que representam um item identificável, uma unidade, uma entidade, real ou abstrata, que desempenha um papel bem definido no domínio apresentado. Em termos mais gerais, um objeto é descrito como algo que possui uma fronteira bem definida. Um objeto é caracterizado por seus atributos e por sua operação (ou operações).

Um sistema complexo é formado por entidades inter-relacionadas, formando uma hierarquia de objetos como mostrado na figura 7. A hierarquia apresenta uma relação biunívoca entre os elementos do problema e sua representação computacional.

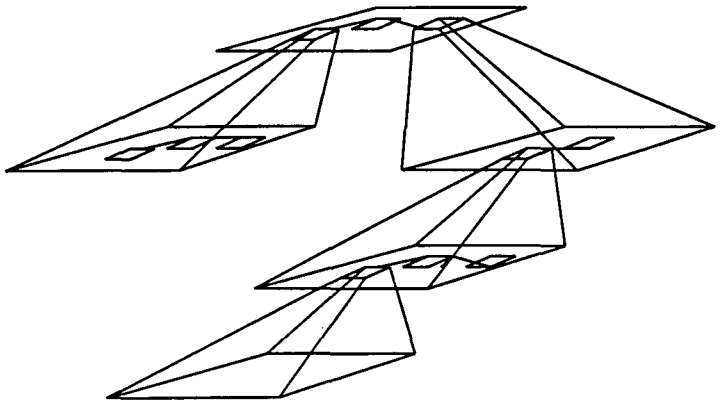


Figura 7. Hierarquia de objetos em um sistema.

Segundo esta relação biunívoca, cada elemento pode se relacionar com os demais de duas formas: por *uso* ou por ***containing***, descritas abaixo e ilustradas na figura 8.

a) Relação de uso

A relação de uso implica na capacidade de troca de mensagens (figura 8 parte 1) entre os objetos, geralmente unidirecional. Cada objeto pode operar como:

- *Atuador* ou objeto ativo, pode operar sobre outros objetos, mas nunca é operado por outro;

- *Agente* ou intermediador de mensagens, atua nos dois sentidos, ou seja, pode tanto operar sobre, como ser operado por outros objetos;
- *Servidor*, nunca opera sobre outro objeto, pode somente ser operado por outros objetos.

b) Relação "containing"

Uma classe **container** (figura 8 parte 2), é uma classe cuja instância é uma coleção de outros objetos, de outra forma, é um objeto que contém outros objetos, porém, estes não são visíveis exteriormente. Um **container** pode denotar uma coleção de objetos *homogênea* (todos os objetos nesta coleção são da mesma classe) ou *heterogênea* (cada objeto pode ser de classes diferentes, ainda que alguns compartilhem uma superclasse comum).

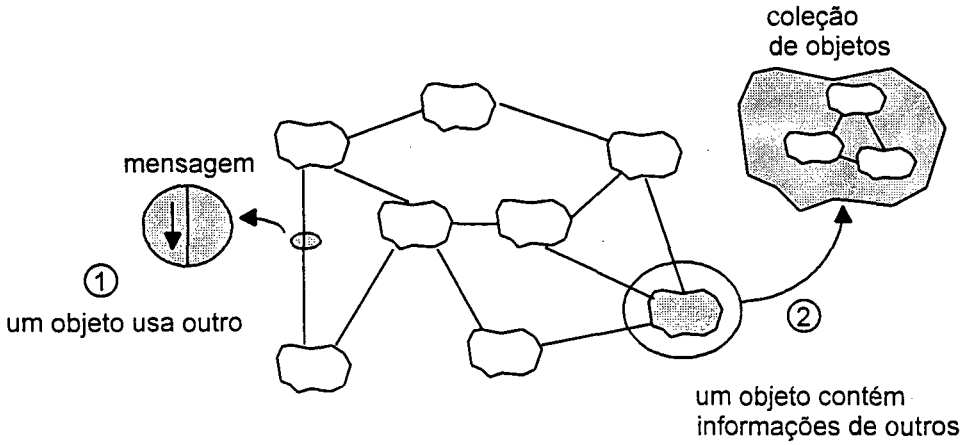


Figura 8. Relação entre objetos

Rumbaugh [50], define uma *ligação* como uma conexão física ou conceitual entre dois objetos, e *associação*, um conjunto destas ligações e Embley [18] define esta relação como "membro de (**is member of**)". Rumbaugh ressalta ainda que, pode ser útil modelar uma classe como uma associação, e cada ligação se torna uma instância desta classe.

Neste contexto, a agregação é uma relação do tipo "parte de (**a part of**)" [18], [50], e suas propriedades básicas são:

- a *transitividade* que diz, se A é parte de B e B é parte de C, então A é parte de C;

- e o *antissimetrismo* que diz, se A é parte de B, então B não é parte de A.

3.2. CLASSES

A observação dos objetos envolvidos no problema, leva a classificação destes, segundo propriedades (atributos) e comportamentos (funções) comuns. As características comuns, abstraídas dos objetos a partir da classificação, reúne estes em uma classes de objetos comuns.

Uma classe contém estruturas de dados e funções:

- as estruturas de dados compõem os atributos, que descrevem as características comuns;
- as funções (ou ainda métodos ou operações [7]) definem o comportamento de um objeto, e podem:
  - pertencer somente a classe (métodos encapsulados);
  - formar a interface da classe (métodos públicos), ativados por mensagens.

O único método da classe que pode ser ativado é o construtor. Através deste método são criados os objetos. Objetos de uma classe possuem atributos que assumem valores diferentes através de parametrização, porém, todos os objetos instanciados possuem os mesmos métodos, que são ativados através de mensagens em tempo de execução.

3.3. HERANÇA

A herança é o principal mecanismo de reutilização de código em orientação a objeto, podendo ser utilizada tanto para expressar *generalização* como *especialização* (figura 9).

A herança caracteriza uma hierarquia de classes "tipo de (**kind of**)". Esta hierarquia é construída em função dos vários tipos encontrados no problema. Os

relacionamentos são construídos em função da reutilização de características em comum na construção de novos tipos. De maneira simplificada, uma classe implementa um tipo.

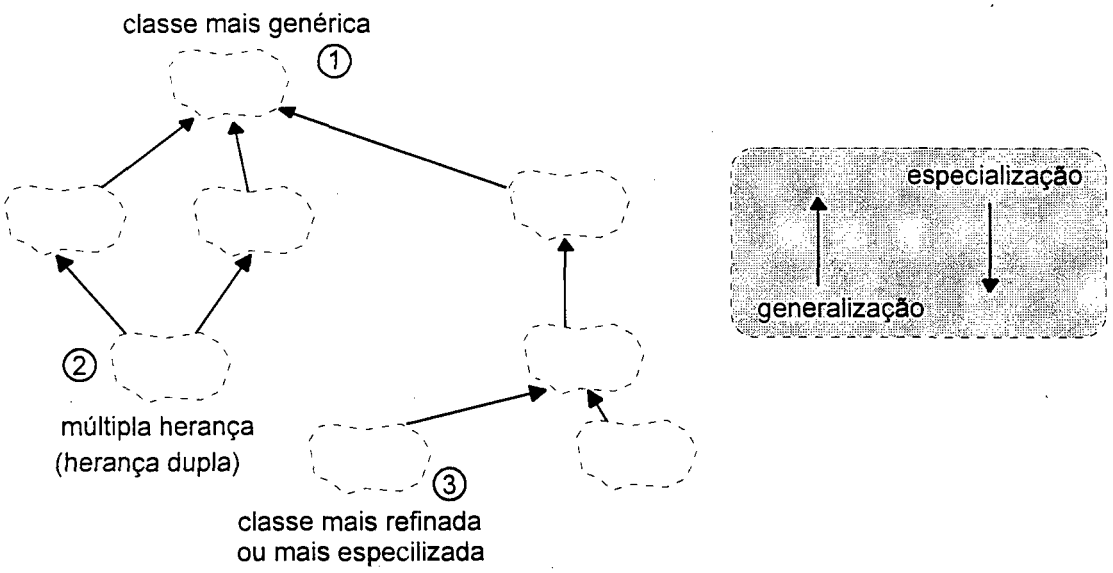


Figura 9. Hierarquia de classes.

A classe mais genérica (1), contém em geral, um corpo básico para as outras classes. As subclasses ou classes herdeiras formam uma especialização da classe base, usando dados e métodos da classe superior, e incrementando seus próprios dados e métodos. As especializações podem ser por herança simples (3) ou múltiplas (2).

A classe fornece a possibilidade de construir módulos fracamente acoplados, encapsulando detalhes que não contribuem para os relacionamentos entre objetos, desta forma restringindo ao seu interior os efeitos causados por modificações.

3.4. POLIMORFISMO

Com o polimorfismo, denomina-se a característica, de que uma determinada mensagem possui interpretação diferente para objetos pertencentes a classes diferentes, ou seja, a mesma mensagem ativa métodos diferentes.

Um exemplo de hierarquia com polimorfismo é mostrado na figura 10. Este exemplo trata de dispositivos eletrônicos com algumas características em comum, porém, com funções (ou operações) diferentes. As características comuns são reunidas no fato de todos possuírem *entradas*, *saídas* e uma *operação*. Esta operação transforma os dados recolhidos na entrada e apresenta o resultado da transformação na saída.

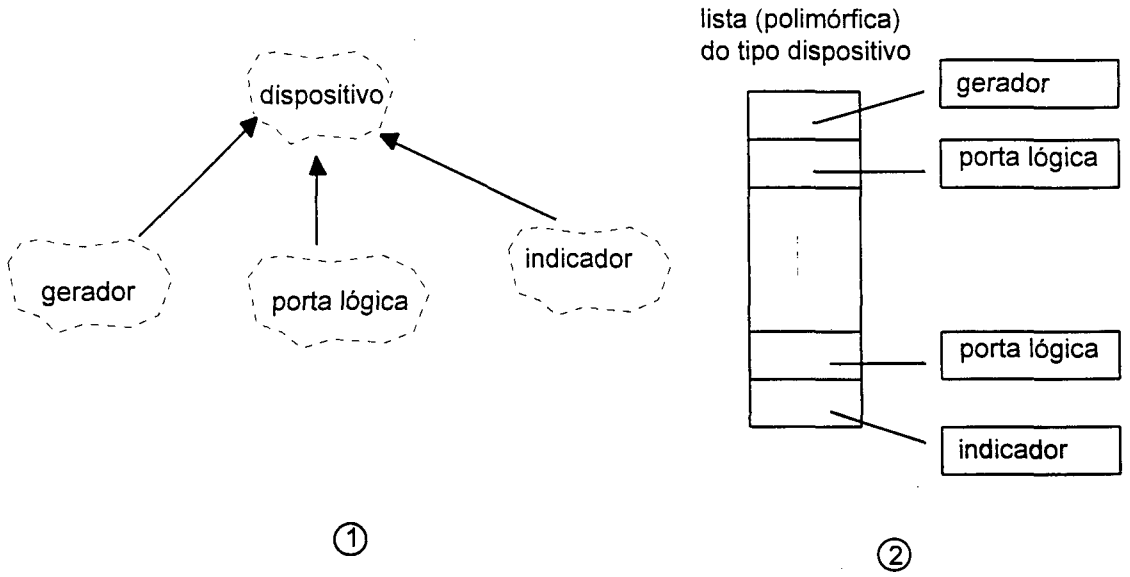


Figura 10. Exemplo de uma possível hierarquia de classes com polimorfismo.

Assumindo que existem dispositivos com funções lógicas distintas, a classe base poderia conter os seguintes métodos:

- *ajustaEntrada*, que atualiza as entradas com dados atuais;
- *transformaDados*, que efetua a transformação sobre os dados da entrada;
- *ajustaSaída*, atualiza a saída.

Todos os objetos, receberão a mensagem *transfomaDados*, porém, cada um a executará de acordo com sua implementação.

3.5. ANÁLISE ORIENTADA A OBJETOS

É o primeiro passo da metodologia orientada a objetos, deve-se aqui criar um modelo preciso do mundo real ou do problema técnico. Antes de se construir um sistema de **software** e **hardware**, ou mesmo outro sistema complexo, com conotação diferente, o construtor deve interpretar e entender muito bem os requerimentos e o ambiente físico real no qual o sistema existe.

O segundo propósito da análise orientada a objetos, é modelar o sistema físico de forma a torná-lo compreensível, identificando os objetos do problema. Para tal, é preciso primeiro abstrair as características importantes dos objetos, e depois os detalhes diferenciadores. A partir desta abstração, poder-se-á reunir os objetos com características comuns em uma classe, e a partir dos detalhes diferenciadores criar especializações desta.

A análise é feita evitando decisões de implementação, ou seja, o seu resultado deveria ser: entender o problema e prepará-lo para o projeto.

3.6. PROJETO ORIENTADO A OBJETOS

Após a análise do problema, deve-se decidir como projetar. Rumbaugh [50], divide o projeto em dois: o projeto do sistema e o projeto dos objetos.

O primeiro, é uma estratégia de alto nível para resolver o problema e construir a solução. Aqui são tomadas decisões sobre a divisão (ou organização) do sistema em subsistemas, sobre os componentes necessários e sobre o curso que a construção do sistema deve seguir. Em outras palavras, define-se aqui o plano de ataque. Esta fase é muitas vezes fundida com o segundo propósito da análise

O segundo, define completamente as classes e suas relações a serem aplicadas na implementação, bem como as interfaces e algorítmos dos métodos utilizados para implementar as operações.

3.7. IMPLEMENTAÇÃO ORIENTADA A OBJETOS

Mesmo que não obrigatório, deve-se utilizar aqui uma linguagem apropriada, pois, é fácil verificar que as linguagens convencionais não suportam diretamente, vários mecanismos que caracterizam a orientação a objetos, e uma linguagem orientada a objetos facilita a implementação do modelo projetado baseado no paradigma da orientação a objetos.

No método da programação orientada a objetos, os programas são organizados em conjuntos de objetos cooperantes, cada qual representando uma instância de uma classe, que são membros de uma hierarquia de classes unidas por algum tipo de relação.

Dentre as muitas vantagens na utilização de uma linguagem orientada a objetos, está o fato da linguagem ter sido desenvolvida visando:

- Facilitar a implementação de estruturas mais complexas e adicioná-las à bibliotecas.
- Permitir a reutilização de código. Se uma classe da biblioteca não satisfaz a necessidade corrente, é possível alterar ou estender uma parte desta, sem que seja preciso conhecer detalhes de seu funcionamento interno.
- Desenvolver a manutenção de código. Na linguagem que suporta o conceito de orientação a objeto, o código é geralmente mais fácil de ser entendido, consertado ou modificado.
- Encapsulamento. A modificação de um determinado componente do sistema não acarreta conseqüências imprevisíveis em outras partes, o que facilita a depuração de programas.
- A passagem do projeto à implementação é direta, pois a linguagem foi construída sob o mesmo paradigma.

Entre as linguagens de programação construídas sob o paradigma da orientação à objetos destaca-se o SmallTalk e o C++, principalmente pela disponibilidade no mercado. Pesquisou-se as estas duas linguagens, e adotou-se o C++ pelas seguintes razões:

- é uma linguagem híbrida, que permite a programação orientada a objetos mas que, para funções críticas, possibilita a programação procedural. Característica muito útil, nos casos que envolvem sistemas de tempo real.

- É uma linguagem compilável, ou seja, gera um código executável, facilitando assim o transporte do sistema, permitindo ao público alvo (o aluno) facilidades no sentido de possuir uma cópia do programa em disco, que poderá ser utilizada em qualquer computador.

4. ANÁLISE DO PROBLEMA

A análise do problema é feita em duas fases: a análise dos sistemas físicos existentes e a análise destes sistemas identificando os objetos e suas classes.

4.1. ANÁLISE TÉCNICA COM BASE NA TEORIA DE SISTEMAS

O simulador a ser desenvolvido deve descrever o comportamento dinâmico de diferentes sistemas físicos. Uma base adequada para modelagem e descrição destes sistemas, para posterior implementação em um programa de computador, encontra-se na teoria de sistemas.

A seguir, apresenta-se os principais e relevantes aspectos desta teoria para o ara presente trabalho, tendo em vista uma possível implementação de um simulador conforme o paradigma orientado à objetos.

4.1.1. MODELAGEM DE SISTEMAS

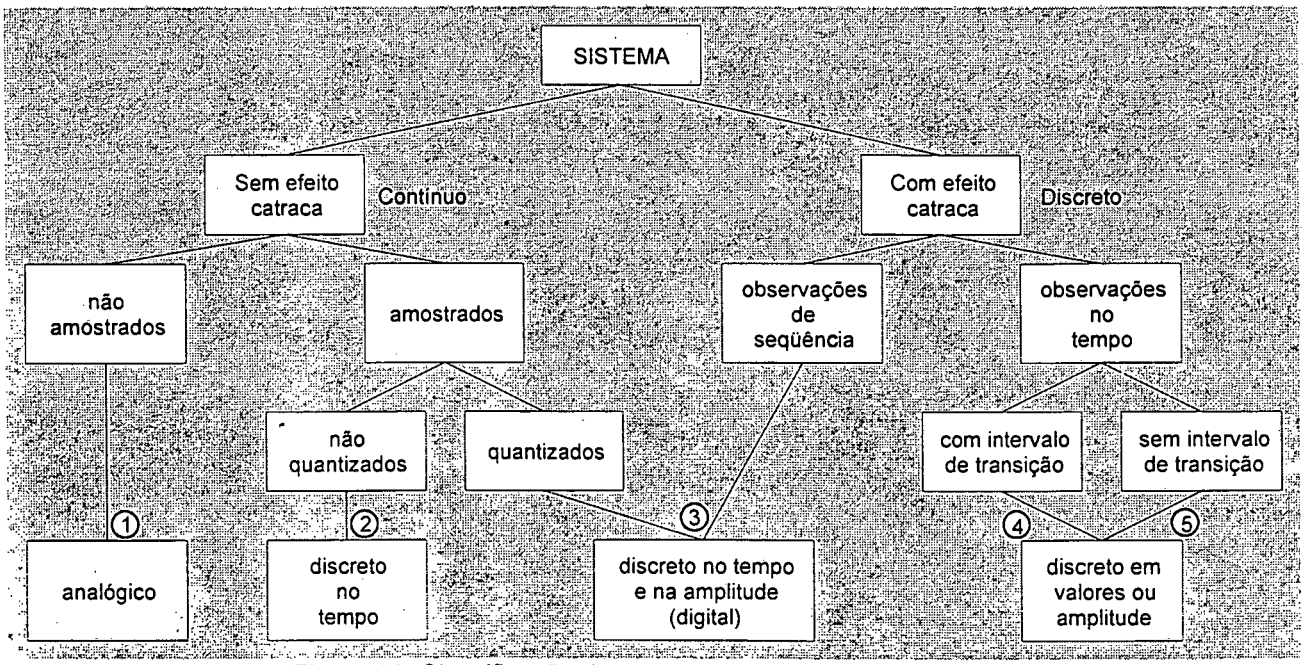


Figura 11. Classificação do comportamento de sistemas.

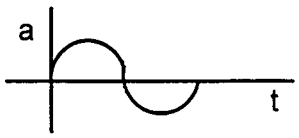
Um sistema pode ser definido como um conjunto de elementos interligados conforme uma estrutura. O objetivo de um sistema é de transformar matéria, energia ou informações.

A análise de sistemas, estuda o comportamento futuro de sistemas reais, enquanto que a síntese, com a determinação do comportamento futuro de sistemas a serem projetados. A análise e a síntese são feitas através de modelos que descrevem de forma adequada, o comportamento de um sistema físico.

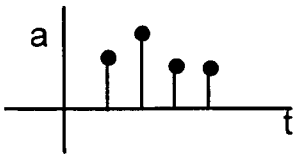
Os sistemas podem ser classificados através do seu comportamento. Na figura 11 é mostrado esta classificação hierarquicamente Wendt [60].

Pode-se observar ao final dessa hierarquia, que conforme seu comportamento, os sistemas se dividem em cinco modelos:

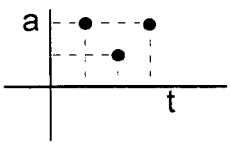
- *sistemas analógicos* (1), podem assumir qualquer valor tanto no tempo como na amplitude. Como exemplo tem-se, os sons, a pressão atmosférica e a corrente elétrica;



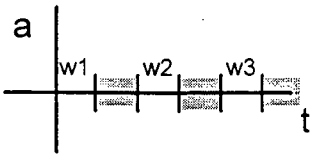
- *sistemas discretos no tempo e contínuos na amplitude* (2), podem assumir qualquer valor na amplitude, porém, são analisados de tempo em tempo discretamente. O estoque semanal, o estudo demográfico e a temperatura média diária, são exemplos deste modelo;



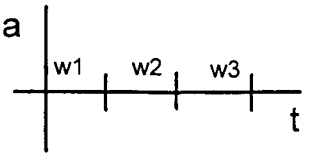
- *sistemas discretos no tempo e amplitude* (3), assumem somente valores dentro de um conjunto limitado de possibilidades, e são regidos por um relógio, onde, a mudança de estado é feita apenas em tempos específicos. Como exemplo, cita-se a catraca e os geradores de onda quadrada;



- *sistemas contínuos no tempo e discretos na amplitude* (com intervalo de transição) (4), os valores aqui assumidos, não são regidos por um relógio, ou seja, podem mudar a qualquer instante, e essa mudança necessita de um intervalo de tempo para sua efetivação (ex. placar);



- *sistemas contínuos no tempo e discretos na amplitude* (sem intervalo de transição) (5), da mesma forma que o anterior, porém a mudança de estado é imediata, por exemplo o estado civil de um cidadão.



Um resumo dos possíveis modelos de sistemas, é apresentado na tabela abaixo, que é formada por todas as combinações entre as abordagens de tempo e amplitude. As linhas da primeira coluna (modelo), correspondem aos números utilizados nos modelos da figura 13.

modelo	amplitude	tempo
(1)	contínuo	contínuo
(2)	contínuo	discreto
(4) e (5)	discreto	contínuo
(3)	discreto	discreto

Tabela 1: Possíveis tipos de sistemas físicos

Na realidade, encontra-se na natureza, apenas sistemas contínuos no tempo e amplitude (1), porém, na prática computacional digital trabalha-se apenas com modelos discretos no tempo e amplitude (3).

4.1.2. ESTRUTURA DOS COMPONENTES

Na teoria de sistemas costuma-se representar os elementos dos sistemas através de caixas, a estrutura do sistema através de ligações e a atividade da caixa por operações.

A figura 12 apresenta uma caixa com entradas (E), operação (Op) e saídas (S).

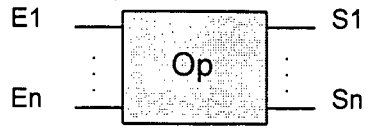


Figura 12. Caixa que representa um elemento ativo

A atividade da caixa pode então, ser descrita pela operação (Op) mostrada na equação 1, que em geral consiste em uma função matemática (equação 2) efetuada sobre os dados de entrada (E1..En), gerando então, os dados das saídas (S1..Sn).

$$\vec{S} = Op(\vec{E}) \tag{1}$$

$$\vec{S} = F(\vec{E}) \tag{2}$$

Definida a operação ou função, correspondentemente a essa, tem-se um elemento mínimo, representado por funções elementares algébricas ou lógicas (booleanas).

Um sistema (figura 13) é resultado da interligação de vários elementos mínimos em série (1), paralelo (2) ou com realimentação (3). Este sistema pode ainda ser visto como um único elemento (4) com entrada (E) e saída (S).

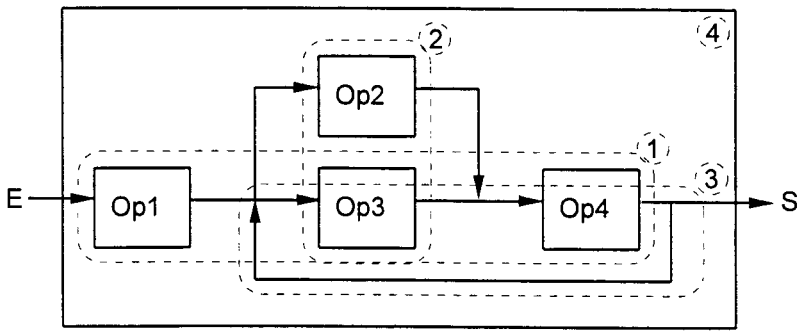


Figura 13. Exemplo de um sistema com vários elementos

Na eletrônica, sistemas conforme a figura 13, são denominados também de circuitos, e os dois termos podem ser utilizados como sinônimos.

Um sistema físico interage com o ambiente recebendo alimentação ou estímulo nas suas entradas, e gerando reações observáveis nas suas saídas, existem portanto, unidades geradoras de sinais e unidades indicadoras dos sinais.

Os geradores, geram sinais com base em uma operação embutida, parametrizável pelo usuário do simulador. Os geradores de sinais, são ligadas com o sistema a simular através de suas saídas, como representado na figura 14.

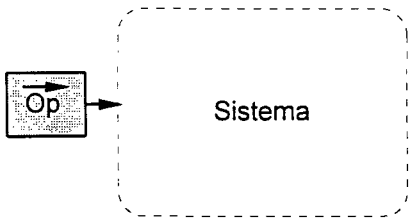


Figura 14. Exemplo de um sistema com entradas geradoras alimentando o circuito.

Os indicadores de sinais, representados na figura 15, são ligadas ao sistema através de suas entradas, e apresentam os resultados deste. São também baseadas em uma operação embutida e parametrizável.

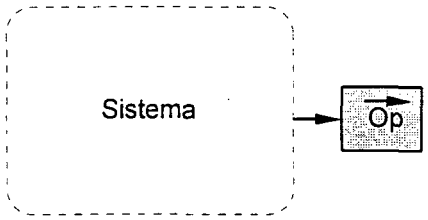


Figura 15. Exemplo de um sistema com saídas indicadoras mostrando o resultado do sistema.

Num simulador, os geradores e indicadores são implementados por **software**, porém, pode-se utilizar geradores e indicadores reais (**hardware**), acessados através de interfaces físicas.

Neste caso os sinais do/para o ambiente externo ao simulador, são postos a disposição do simulador por funções de leitura e escrita em portas da interface.

4.1.3. AS FUNÇÕES DOS COMPONENTES

Como discutido nos itens anteriores, sistemas podem ser vistos como caixas que contém um conjunto de entradas e um de saídas. Dentro da modelagem descrita no item 4.1, estes circuitos são divididos segundo seu comportamento, ou seja, de acordo com suas funções, que são basicamente discretas e contínuas.

Do ponto de vista da eletrônica, dispositivos podem ser respectivamente *digitais* (combinacionais, seqüenciais) ou *analógicos*.

a) SISTEMAS DIGITAIS COMBINACIONAIS

Sistemas digitais combinacionais, são conhecidos na eletrônica, como circuitos combinacionais, e eles são tais que suas saídas dependem unicamente das entradas independentemente do tempo, de outra forma, todas as entradas são supridas por fontes externas e podem assumir somente dois valores de amplitude (binários). Esta definição se enquadra no modelo *sistemas discretos no tempo e amplitude*.

Na lógica digital, funções mínimas, são operações Booleanas com uma entrada (*inversor*) ou duas entradas (*E* e *OU*) e uma saída.

Circuitos Combinacionais, podem ser completamente representados por:

- expressões booleanas;
- tabelas verdade;
- símbolos gráficos.

Todo circuito combinacional pode ser construído a partir *portas lógicas elementares* ou *funções mínimas*, e mais, um circuito combinacional com uma saída somente, é considerado uma *porta lógica*.

A figura 16, mostra um circuito combinacional, que pode ser visto: em detalhes (1), onde, as caixas são as próprias portas lógicas mínimas que representam funções booleanas;

através de apenas uma caixa (2) que efetua uma transformação sobre as entradas, equivalente à função do circuito, e finalmente através de sua tabela verdade (3).

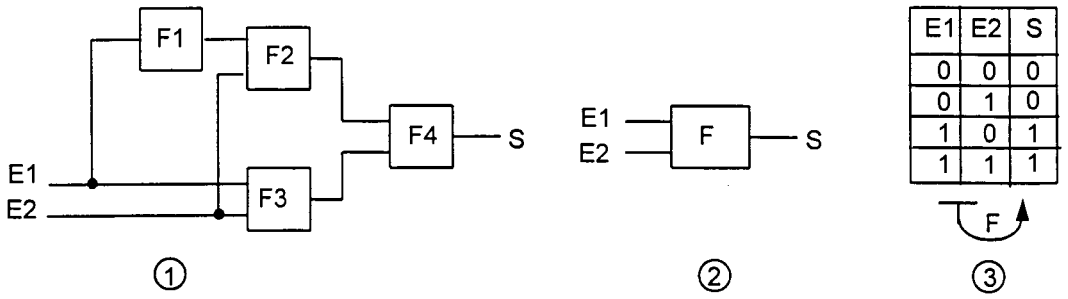


Figura 16. Ex. de um Circuito Combinacional (1), a caixa que o representa (2) e sua tabela verdade(3)

Freqüentemente, encontra-se na bibliografia, a atribuição de *blocos funcionais* para as caixas que representam circuitos combinacionais que são formados por mais de uma porta lógica, e neste texto atribui-se o termo *macro-funções*.

Dentre estas macro-funções pode-se destacar:

- Decodificadores;
- Codificadores;
- Multiplexadores;
- Demultiplexadores;
- Comparadores;
- Dispositivos aritméticos.

Através da análise de circuitos combinacionais, verifica-se que estes dispositivos seguem ao princípio das caixas com entrada, função e saída, porém, em muitos deles, especialmente os que tem macro-funções, encontra-se entradas com funcionalidades adicionais às executadas pelas entradas de dados. Estas entradas controlam as funções dos dispositivos, mantendo porém, a característica básica de uma entrada: receber informações.

A figura 17 mostra na sua primeira parte, um dispositivo genérico (1) com entradas de dados e habilitação.

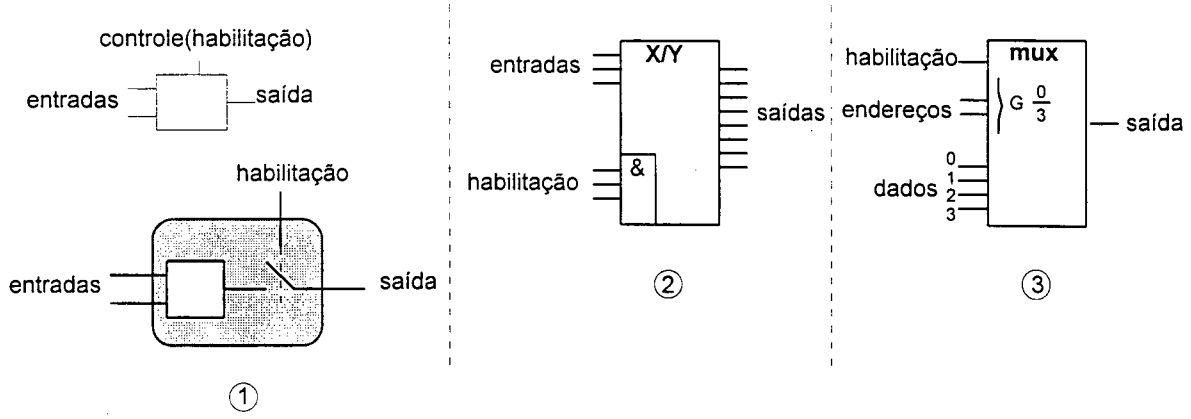


Figura 17. Um dispositivo genérico (1) com terminais para habilitação, um decodificador 3x8 (2), e um multiplexador 4x1(3).

Como observa-se, quando o terminal de habilitação está desativado a saída não recebe nenhum dos dois níveis lógicos do sistema binário, mas um *terceiro estado* diferente dos dois. Na segunda parte é mostrada o exemplo de um decodificador (2) com três entradas, oito saídas e um terminal de habilitação. E na terceira parte um multiplexador com quatro entradas, uma saída e dois terminais de endereço, que determinam o número da entrada (0 a 3) que é copiada na saída.

b) SISTEMAS DIGITAIS SEQÜENCIAIS

Sistemas digitais seqüenciais, são conhecidos na eletrônica como circuitos seqüenciais. Nesta área, o que define as saídas, são as entradas presentes, e a história das entradas anteriores. Em outras palavras, as saídas dependem da seqüência das entradas anteriores até o estado presente, daí o nome seqüencial. Na modelagem apresentada no item 4.1, este tipo de circuito se enquadra na definição de *sistemas discretos no tempo e amplitude*.

Neste tipo de circuito, é somado um novo conceito às operações ou funções mínimas: a *memória* (figura 18). É um elemento que *armazena*, em geral sincronamente com um relógio, os dados recebidos em sua entrada, e os deixa disponível em sua saída.

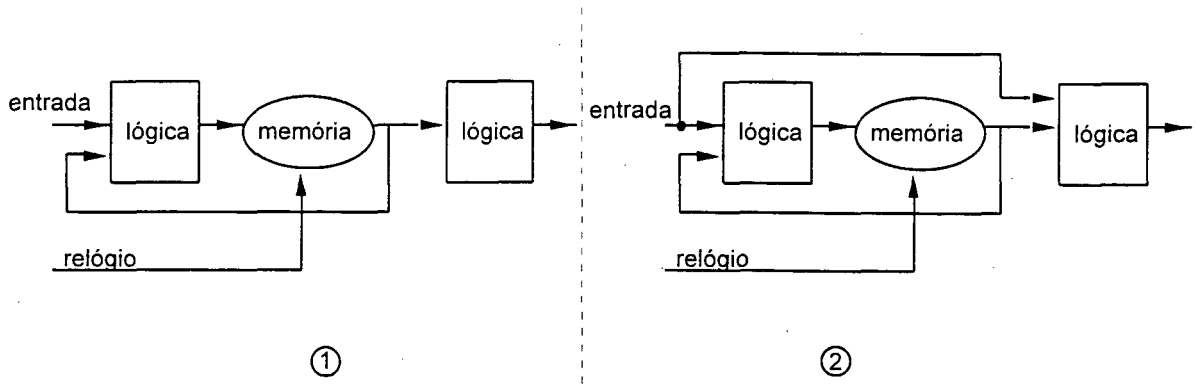


Figura 18. Exemplo de uma estrutura genérica representando um circuito seqüencial de Moore (1) e de Mealy (2).

Um circuito em que as saídas são funções apenas do estado ou dos dados armazenados, é chamado de *circuito de Moore* representado na figura 18 parte 1, e um circuito em que as saídas são funções tanto do estado quanto das entradas é chamado de *circuito de Mealy* (figura 18 parte 2).

Existem também os circuitos seqüenciais assíncronos, porém, menos utilizados na prática, e portanto não é implementado neste trabalho.

Como principais funções mínimas, bases para os circuitos seqüenciais, tem-se os elementos de memória:

- **latch's**;
- **flip-flop's**;

e como macro-funções que representam circuitos seqüenciais básicos destaca-se:

- contadores;
- registros de deslocamento.

Os **latch's** e os **flip-flop's** efetuam o mesmo tipo de transformação sobre os dados, porém se diferenciam pela forma com que os dados são transferidos para a saída, ou seja, o tipo de sincronismo com o relógio:

- os **latch's**, podem possuir controle de habilitação ou não, caso possuam, esta habilitação é efetivada pelo nível (alto ou baixo) do sinal de controle (ou pulso do relógio). Estas característica os denotam como transparentes;
- os **flip-flop's** são sempre controlados por uma linha de habilitação. Este controle poder ser:
  - do tipo mestre-escravo, onde os dados são lidos da entrada a partir transição positiva do relógio, e transferidos para a saída a partir da transição negativa do relógio;
  - ou do tipo gatilhado pela borda, onde os dados são transferidos para a saída exatamente na transição (positiva ou negativa) do relógio.

Como exemplo de elementos mínimos da área em questão, a figura 19, mostra o símbolo de um **latch** set/reset (1) sem controle de sincronismo, de um **flip-flop** D (2) controlado por nível, e um **flip-flop** JK (3) controlado pela borda positiva do relógio.

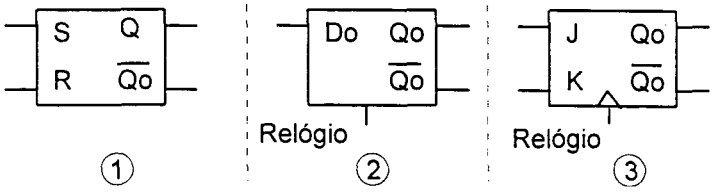


Figura 19. Um **latch set/reset**, e os **flip-flop's** D e JK

Circuitos seqüenciais, em geral, podem ser representados e implementados como máquinas de estados.

Existem várias formas de apresentação de uma máquina de estados, na figura 20 é mostrada a representação gráfica de uma máquina de estado hipotética, e a tabela 2, mostra a mesma máquina de estados na forma de tabela de estados.

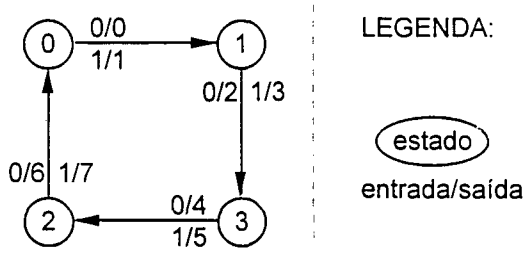


Figura 20. Representação gráfica da máquina de estados.

estado atual		entrada	próximo estado		saída		
0	0	0	0	1	0	0	0
0	0	1	0	1	0	0	1
0	1	0	1	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	1	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	0	1	1	0
1	1	1	0	0	1	1	1

Tabela 2. Uma máquina de estados hipotética representada pela sua tabela de estados.

Em nenhum dos casos é mostrado o controle do relógio, porém, deve ficar claro que as mudanças de estado neste tipo de sistema, acontecem sempre em sincronismo com este.

c) ARQUITETURA DE COMPUTADORES

No contexto deste trabalho, a área arquitetura de computadores, pode ser considerada como um conjunto de bloco funcionais, ou macro-funções digitais iguais às dos blocos funcionais citados nos itens de circuitos digitais combinacionais e seqüenciais.

Uma Unidade Central de Processamento (UCP), que descreve uma macro-função, por exemplo, pode ser composta por dispositivos que descrevem as funções mínimas desta área:

- Unidade Lógica Aritmética (ULA);
- registradores;
- controlador ou unidade de controle (UC);
- contador de programa;
- memória;

Um exemplo para o uso de macro-funções, encontra-se em Malvino [36] que propõe uma CPU suficientemente simples para introduzir o aluno à área. A figura 21, mostra o diagrama de blocos desta CPU, com os seguintes blocos funcionais: contador de programa; entrada e REM, registrador de instruções, controlador; registrador acumulador; somador/subtrator; registrador de uso geral (B), registrador de saída e display (indicador).

O *contador de programa* é inicializado com zero (0000),e incrementado de um em um, assim que uma instrução é trazida da memória para a CPU (opcode-fetch).

O bloco *entrada e REM*, inclui um registrador de dados, utilizado para armazenar o dado que será enviado para RAM, e Registador de Endereço de Memória (REM), armazena o endereço da memória. Este bloco é controlado por chaves externas, que são utilizadas para escrever o programa do computador que será armazenado na RAM.

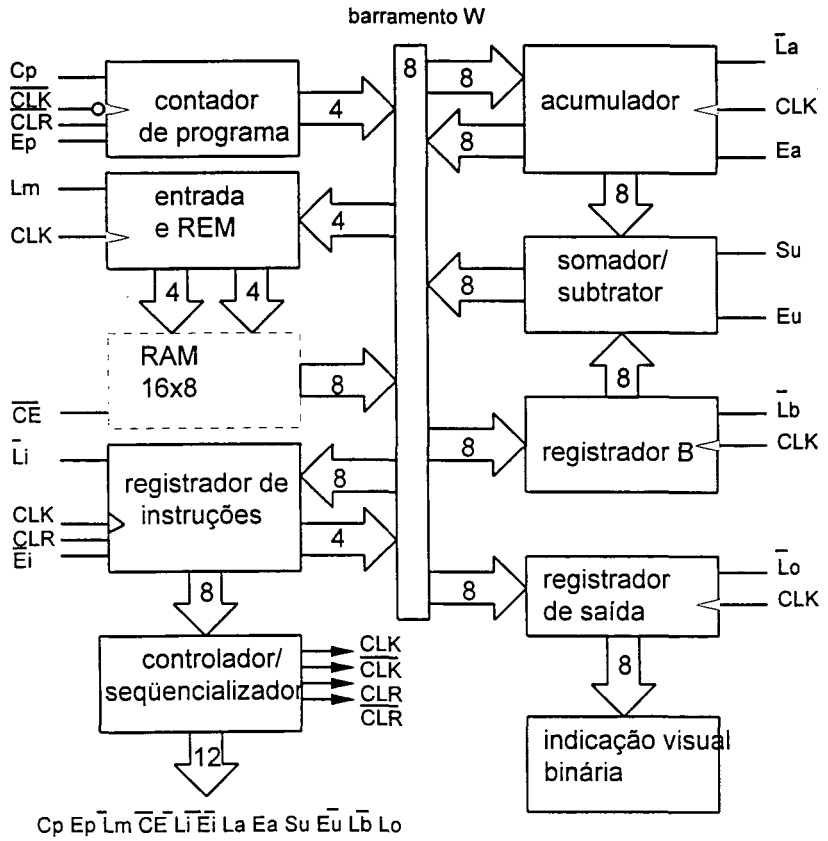


Figura 21. Arquitetura da CPU SAP-1 (reprodução da figura 10-1, pag. 257, [36]).

A *RAM* (**Read Only Memory**) é uma memória estática de dezesseis posições com oito bits cada (16x8).

O *registrador de instruções* constitui parte da unidade de controle, ele recebe a instrução lida na memória.

O *controlador-seqüencializador*, define as ações, ou a seqüência de ações que o computador deve seguir para efetuar as tarefas, em outras palavras, ele gera as microintruções necessária para cada ciclo de execução de uma instrução.

O *acumulador* (A) armazena respostas intermediárias durante o processamento. A saída para o barramento é do tipo três estados, e para o somador/subtrator é biestável.

O *somador-subtrator* efetuas duas operações: S=A+B; e A=A+(-B), onde (-B) é o complemento de 2 de do registrador B.

O *registrador B* é um registrador auxiliar para as operações aritméticas.

O *registrador de saída* contém o dado a ser enviado para fora do computador.

O *indicador visual binário* é uma fileira de oito **led's** que mostra o conteúdo do registrador de saída.

O autor implementa todos estes blocos funcionais, através de circuitos TTL. Uma decomposição dos blocos em elementos mínimos e macro-funções, é mostrada no exemplo da figura 22. Esta figura trata de parte do circuito eletrônico que implementa o computador: o registrador acumulador (A), o registrador B e o somador-subtrator.

O acumulador é composto por dois circuitos integrador (CI's) TTL 74LS173, que são compostos por quatro **flip-flop**'s (biestáveis) cada, totalizando oito linhas de dados. Dois CI's 74LS126, que são **buffers tri-state**, colocam o conteúdo do acumulador no barramento.

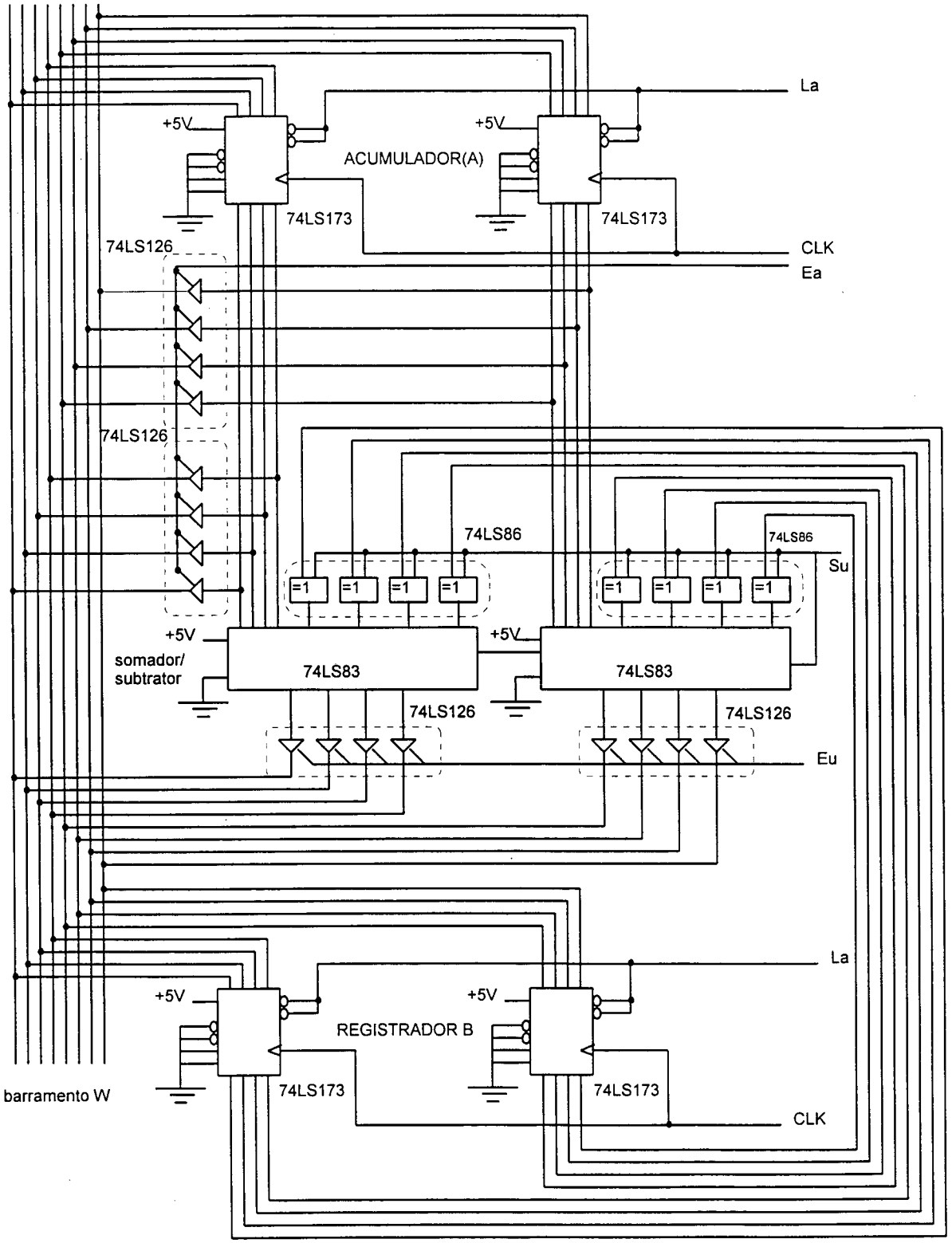


Figura 22. Registradores A e B e somador subtrator (reprodução parcial da fig 10-13, pag. 283/284, [36])

Os dois CI's 74LS86, formam um conjunto de oito portas tipo ou-exclusivo, que controlam o tipo de operação. Se Su é baixo, o conteúdo de B é transferido para o somador, porém, se Su é alto, o complemento de 1 é transmitido e um 1 é acrescentado para formar o complemento de 2. O somador é composto por dois CI's 74LS83.

O registrador B é formados por CI's iguais aos utilizados no acumulador, ele contém os dados a serem somados ou subtraídos do acumulador.

Um segundo exemplo é mostrado na figura 23, onde pode-se ver um conjunto de blocos funcionais formando um controle microprogramado.

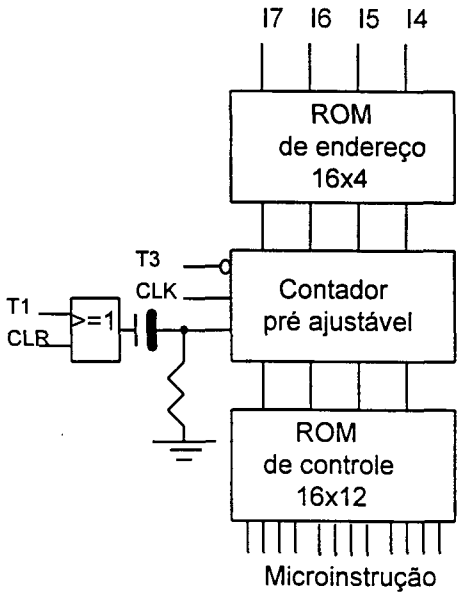


Figura 23. Controle Microprogramado da CPU (reprodução da figura 10-16, pag. 289, [36])

Um controlador ou seqüencializador pode ser construído através da lógica combinacional, porém, com conjuntos de instruções maiores, um controlador exige centenas e até milhares de portas. A microprogramação é uma forma alternativa de produzir palavras de controle que executam o processamento do computador. A idéia básica é armazenar as microintruções em uma ROM.

As microintruções de cada instrução, inclusive a rotina de busca da instrução, são armazenadas na ROM de controle.

A ROM de endereço, contém os endereços de partida de cada rotina armazenada na ROM de controle.

O contador pré-ajustável, recebe o endereço do início da rotina de microinstruções quando T3 é alto, e de forma contrária, ele conta a partir do valor pré-ajustado, ou seja o endereço inicial da rotina.

d) SISTEMAS ANALÓGICOS

Sistemas analógicos, são identificados na eletrônica, como circuitos analógicos, e descrevem *sistemas contínuos no tempo e amplitude*. Como definido no item 4.1, a abordagem informatizada deste tipo de sistema, envolve a discretização dos valores no tempo e amplitude, porém, na discretização de um sistema, devem ser conservadas as características deste sistema, ou em outras palavras, deve-se escolher um intervalo de tempo entre duas amostras (período de amostragem) tão pequeno quanto possível, de modo a garantir que naquele intervalo a função que descreve o comportamento do sistema é interpolável. Após a discretização este sistema passa a corresponder, referindo-se a figura 13, aos *sistemas discretos no tempo e amplitude*.

O comportamento estático e dinâmico de sistemas contínuos em geral, pode ser descrito por equações diferenciais [24]. Para fins de simulação, parece especialmente interessante uma apresentação de sistemas, utilizando equações de estado.

Uma equação diferencial linear com coeficientes constantes, pode ser apresentado na seguinte forma geral: $a_n y^n + a_{n-1} y^{n-1} + \cdots + a_1 \dot{y} + a_0 y + c1 = b_0 x + b_1 \dot{x} + b_m x^m$, ou de outra forma, normalizada com as constantes de tempo (T) como mostrada a equação 3.

$$\frac{d^n y}{dt^n} = \dot{z}_n = -\frac{1}{T_n^n} y - \frac{T_1}{T_n^n} \dot{y} - \frac{T_2^2}{T_n^n} \ddot{y} - \ldots - \frac{T_{n-1}^{n-1}}{T_n^n} y^{n-1} + \frac{1}{T_n^n} x \qquad (3)$$

Integrando a equação diferencial, pode-se escrever as variáveis de estado (Z), que formam um sistema de equações diferenciais de primeira ordem como segue:

$$z_1 = y$$
$$\dot{z}_1 = z_2 = \dot{y}$$
$$\dot{z}_2 = z_3 = \ddot{y}$$

$$\dot{z}_{n-1} = z_n = y^{n-1}$$
$$\dot{z}_n = \frac{1}{T_n^n} z_1 - \frac{1}{T_n^n} z_2 - \cdots - \frac{T_{n-1}^{n-1}}{T_n^n} z_n + \frac{1}{T_n^n} x = y^n$$

A figura 24 mostra uma implementação deste sistema de equações na forma gráfica. Pode-se observar que gera-se a seqüência das variáveis de estado por integração.

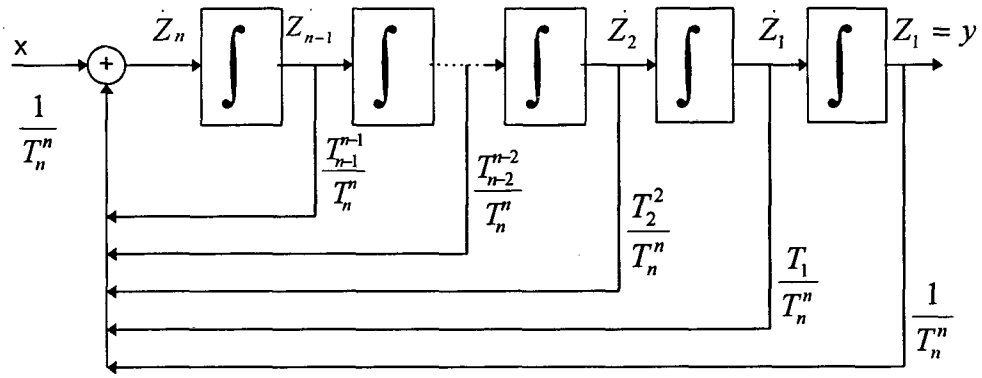


Figura 24. Representação gráfica da forma geral da equação diferencial.

As equações, especialmente na sua forma gráfica, evidenciam que é possível representar qualquer sistema, descritível através de uma equação diferencial (equação 3), apenas com elementos que formam blocos (caixas) com entradas, saídas e funções, interligados entre si. Os elementos mínimos desta área, são implementados pelas seguintes funções algébricas:

- adição;
- subtração;
- multiplicação;

- divisão;

E como macro-função:

- integrador.

Para ilustrar a modelagem de um sistema físico através de uma equação diferencial, passa-se a estudar dois exemplos de processos físicos. São dois exemplos clássicos da literatura e discutidos em cursos da área de controle de processos.

a) Circuito RC

A figura 25 mostra o desenho de um circuito composto, por um resistor (R) e um capacitor (C), denominado de circuito **RC**.

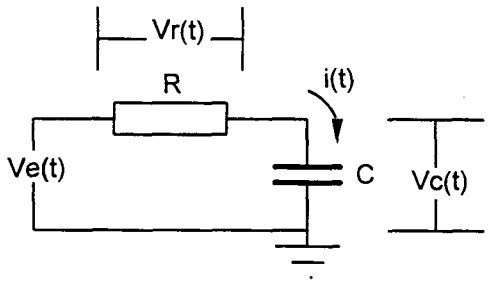


Figura 25. Circuito RC

À entrada do circuito é aplicada uma tensão Ve(t), que é a função de excitação do sistema. Esta tensão, devido às quedas de tensão no circuito, divide-se em Vr(t) que é a queda de tensão no resistor (R) e Vc(t) que é grandeza de estado que se deseja conhecer (saída).

Através da lei de Kirchoff, tem-se $e(t) = Vr(t) + Vc(t)$, e segundo a lei de Ohm $i(t) = \frac{Vr(t)}{\mathrm{R}}$. A corrente que circula no capacitor é dada por $i(t) = C\frac{dVc(t)}{dt}$, e a combinação destas equações resulta em: $\frac{Ve(t) - \mathrm{Vc(t)}}{\mathrm{R}} = C\frac{dVc(t)}{dt}$ ou $Vc(t) = \frac{1}{RC}\int [Ve(t) - Vc(t)]dt$ (4) apresentado também na forma gráfica na figura 26.

Neste exemplo identifica-se que a equação diferencial de primeira ordem apresenta as seguintes operações: soma, multiplicação e integração. Através destes operadores

pode-se simular o comportamento de carga e descarga do capacitor, a partir das condições impostas ao sistema.

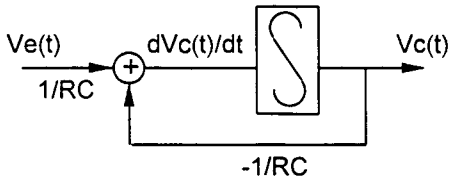


Figura 26. Equação diferencial representada graficamente.

b) Sistema massa/mola/amortecedor

A figura 27, mostra um sistema composto por uma massa, uma mola e uma amortecedor.

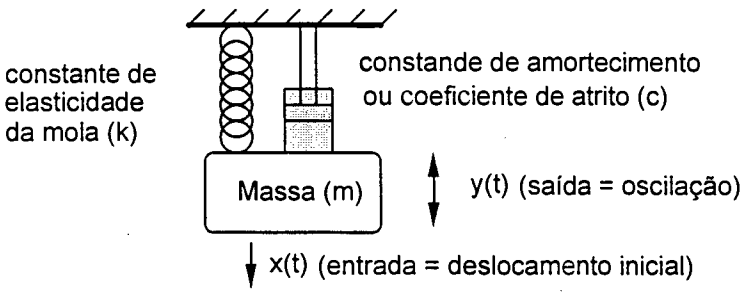


Figura 27 . Sistema massa/mola

De forma resumida, aplicando o princípio de D'Allembert, chega-se a equação 6.

$$\frac{d^2y(t)}{dt} = -\frac{c}{m}\frac{dy(t)}{dt} - \frac{k}{m}y(t) + \frac{1}{m}x(t) \tag{5}$$

$$y(t) = -\frac{c}{m}\int y(t)dt - \frac{k}{m}\iint[y(t) + \frac{1}{m}x(t)]dt \tag{6}$$

e representada pela forma gráfica apresentada na figura 28.

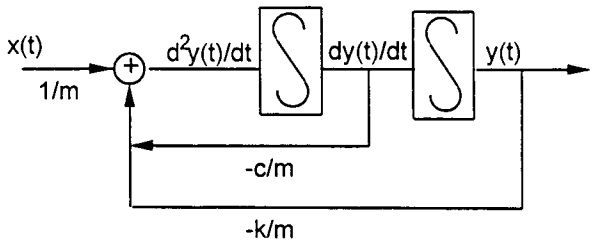


Figura 28. Equação diferencial de segunda ordem para o sistema massa/mola/amortecedor.

Neste exemplo identifica-se, as mesmas operações constantes no circuito RC. Com esta configuração, pode-se simular o movimento da massa (y(t)) segundo os parâmetros da mola (k) e do amortecedor (c) e do impulso inicial (x(t)) aplicado sobre a massa (m).

4.2. IDENTIFICAÇÃO DOS OBJETOS E CLASSES

Como foi dito no item 3.1, as classes abstraem os objetos reais através de características comuns. Nota-se que a teoria de sistemas é uma ferramenta excelente para esta abstração. A sua forma de tratar todos os elementos como sistemas que ao receber um informação em sua entrada, efetua uma transformação sobre ela, deixando o resultado disponível em sua saída, permitiu abstrair as seguintes características comuns entre todos os dispositivos estudados:

- os sistemas são compostos por elementos (caixas), determinados por: entrada; função e saída;
- os sistemas das áreas estudadas, distinguem-se apenas pela função ou operação;
- a construção de um sistema é dada pela interligação entre as entradas e as saídas dos elementos.

Dentro deste contexto, os cinco tipos de sistemas físicos discutidos no capítulo 4, estão de uma certa forma, caracterizados em um único modelo.

Para representar sistemas compostos pelos dispositivos estudados, deve-se ter a priori, uma classe que descreva os dispositivos em si, e uma classe que os reúna através de ligações, formando um circuito.

Refinando-se esta idéia, a classe ***dispositivo*** pode ser construída de duas formas:

- mantendo em seu corpo a estrutura das caixas com entradas e saídas, bem como a função;
- ou mantendo em seu corpo apenas a estrutura com entradas e saídas, comum para todos os objetos. Neste caso, uma outra classe seria

construída contendo as diferentes funções dos objetos, e as duas classes se relacionariam através de uma associação.

Na primeira opção, e necessário criar uma classe para cada tipo de objeto, e na segunda apenas duas classes, esta opção gera menos classes e é utilizada como primeira base para este trabalho.

Uma classe ***circuito*** deverá conter informações sobre os dispositivos existentes, mantidos por exemplo em uma lista, bem como sobre as ligações entre eles.

Para que o usuário possa construir e visualizar os circuitos, faz-se necessário uma classe ***grafica***, que desenhará os dispositivos e suas ligações.

A instalação e parametrização dos dispositivos, e o controle da simulação do circuito, pode ser assumida por uma classe ***menu***.

A notação a ser utilizada para representar relações entre as classes do sistema, é a sugerida por Rumbaugh [50] e mostrada resumidamente nas tabelas 3 e 4.

REGRA	SÍMBOLO
exatamente um	classe
alguns (zero ou mais)	classe
opcional (zero ou um)	classe
um ou mais	1+ classe
especificado numericamente	1-2,4 classe

Tabela 3. Simbologia das regras do relacionamento entre classes.

RELAÇÃO	SÍMBOLO
Associação	classe 1 — nome da associação / regra1 regra 2 — classe 2
Agregação	classe — classe "parte de"
Generalização	superclasse; subclasse 1; subclasse 2

Tabela 4. Simbologia do relacionamento entre as classes.

A figura 29 mostra uma visão global das classes previstas com as devidas relações.

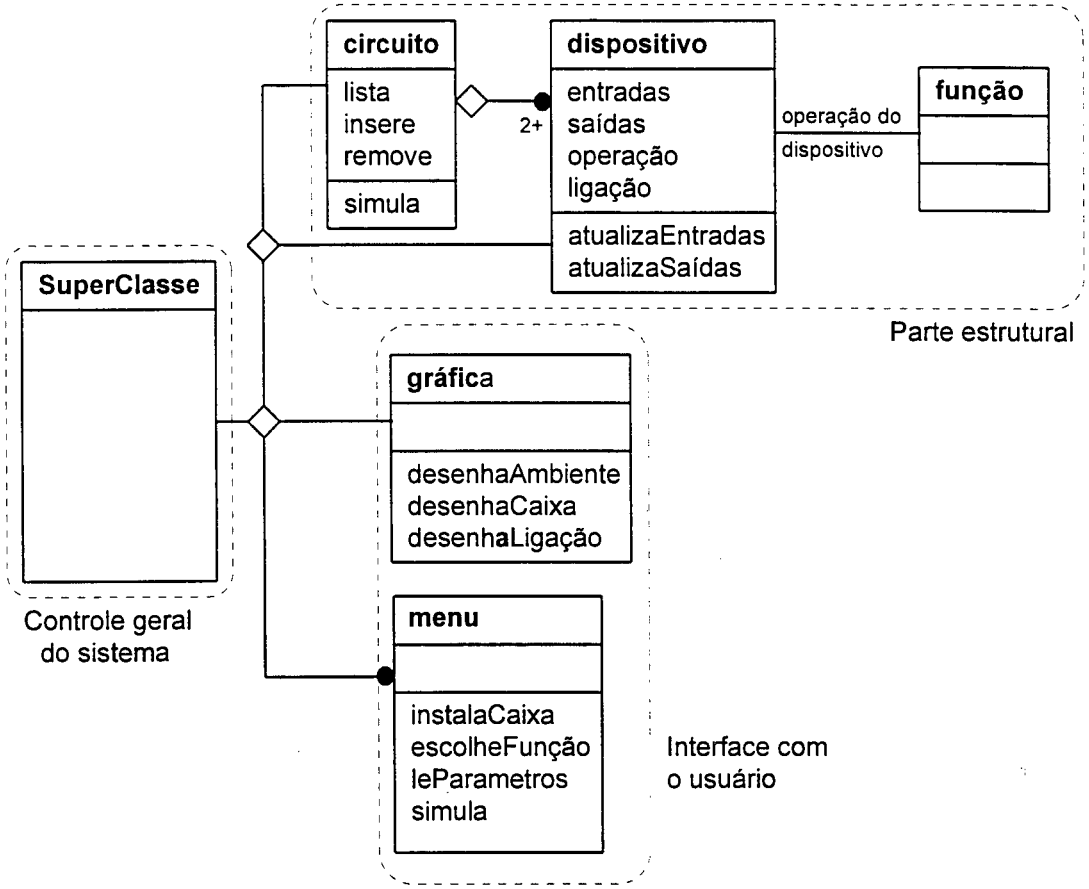


Figura 29. Visão geral do sistema divido em subsistemas.

As possíveis classes apresentadas, até agora descrevem os atributos e a estrutura do simulador estaticamente. Já o comportamento dinâmico do simulador, que envolve a sua própria instalação, a instalação dos componentes e a coordenação das suas atividades (simulação), devem ser assumidos por um gestor, que no caso da figura 29 é indicado como uma superclasse sem considerar ainda os detalhes de implementação.

A responsabilidade desta superclasse é de controlar a execução do programa através de teclas pré-definidas, solicitadas através de menus disponíveis ao usuário.

A idéia básica é construir uma estrutura de seleção dentro de um laço (**loop**) principal, de forma que o fluxo do programa entrasse nele logo no início da execução. Esta estrutura seria então, controlada ao comando do usuário através do teclado.

Supondo-se, por exemplo, o fluxo do programa dentro do laço e aguardando a intervenção do usuário. A tecla **enter** sendo digitada, a superclasse envia as seguintes mensagens:

i. para a classe *gráfica* desenhar a caixa;

ii. para a classe *menu*, desenhar um menu oferecendo as possibilidades de parametrização;

iii. para a classe *dispositivo* instanciar um objeto e efetuar a ligação com seu comportamento;

iv. para a classe circuito inserí-lo na lista.

Após este ciclo, o programa retorna ao fluxo normal, ou seja, ao início do laço onde aguarda nova intervenção.

4.2.1. RELAÇÃO DAS CLASSES COM O CICLO DO COMPORTAMENTO DE UM SIMULADOR

No item 1.3, foi feita uma divisão das tarefas de um simulador. Visando a seqüencialidade então proposta, passa-se a uma descrição do relacionamento das classes projetadas na ordem estudada no capítulo 1.

A primeira parte *introdução*, estará contida na classe ***menu***, através dela o usuário poderá invocar um menu contendo informações para ajudá-lo construir e utilizar um circuito.

O *cenário*, construído com apoio da classe ***gráfica***, constitui-se da visualização do circuito.

A *inicialização da ação*, é dada por parte da classe ***circuito***, quando da verificação e otimização do circuito.

A *ação*, que é a simulação em si, onde a classe ***circuito*** opera sobre ***dispositivos***.

A *reação*, é visualizada com o contribuição da classe ***gráfica***.

A *finalização*, é efetuada pela conjunção de vários métodos como: parar a simulação, salvar o circuito e encerrar o programa. Estas tarefas, são encontradas principalmente na classe ***gráfica***, que deverá reter as informações necessárias para salvamento e reconstrução de um circuito armazenado em disco.

Com isto fecha-se o ciclo do comportamento de um simulador segundo Bodendorf [5].

5. O PROJETO DAS CLASSES E SUAS RELAÇÕES

No projeto, as classes identificadas na análise orientada a objetos (figura 29), são reavaliadas visando um maior detalhamento, baseando-se ainda na análise técnica, porém, considerando possíveis formas de implementação.

Foi visto que existem duas formas básicas de implementar as funções dos dispositivos. Isto pode ser feito no próprio corpo da classe ***dispositivo*** ou utilizando uma outra classe, que se associa a cada objeto, uma função definida na classe ***funções***.

Esta segunda opção traz a vantagem de reduzir o número de classes com o resultado de se ter uma sistema menos complexo e assim com maiores possibilidades de ampliação e manutenção do código.

O maior obstáculo para esta concepção encontra-se na variedade de tipos de funções e dos dados por ela tratados.

Os tipos básicos são os digitais e os analógicos: os primeiros possuem funções booleanas e tratam dados binários; os segundos tem funções algébricas e dados em ponto flutuante.

Analisando mais profundamente as possibilidades de implementação, visualiza-se a opção de implementação por algoritmos tradicionais (como utilizado por Bonatti [6]), ou por tabelas verdade.

A tabela verdade oferece maior flexibilidade e facilidade na implementação de macro-funções e dispositivos definidos pelo usuário, tarefa feita tradicionalmente nos projetos de circuitos digitais.

A figura 30, mostra o relacionamento das classes na forma a ser implementada.

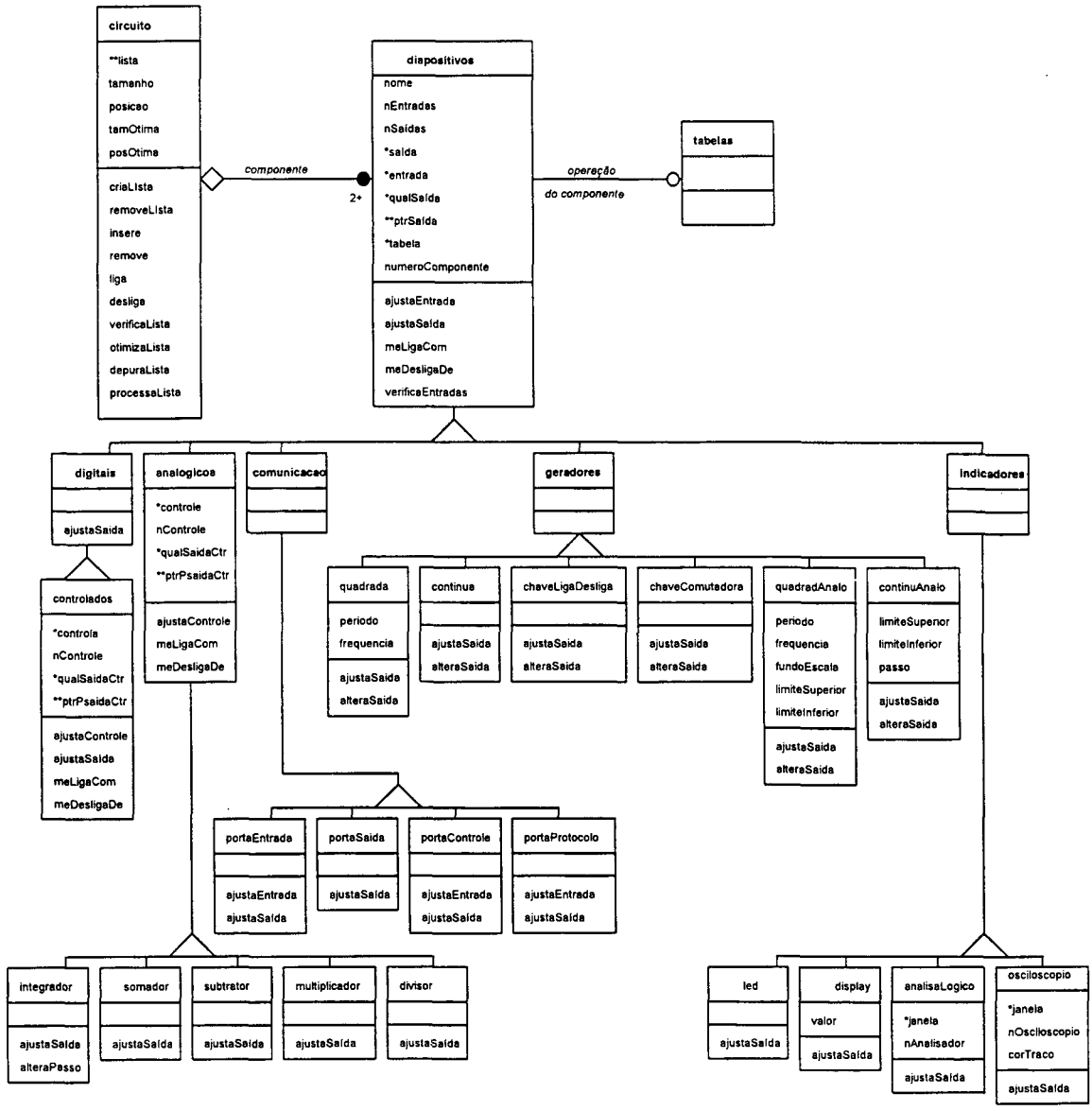


Figura 30. Relacionamento entre as classes : ***dispositivos***, ***tabelas*** e ***circuito***.

Optou-se por utilizar as tabelas, e como conseqüência, tornou-se necessário criar uma estrutura específica, a classe ***tabela*** é resultado do método adotado.

Para suprir a classe ***analogicos***, o ideal seria a construção de uma classe funções analógicas, para agrupar todas as funções dos dispositivos analógicos, porém, por dificuldades encontradas na implementação de testes, os dispositivos desta área são implementados com a função agregada ao corpo da classe.

A ramificação iniciada pela classe ***digitais***, representa, principalmente as portas lógicas, mas também todos os outros dispositivos digitais sem terminais controle. Para representar os dispositivos digitais com controle, cria-se a classe ***controladas***.

A classe ***comunicação***, descreve as portas de comunicação entre o simulador e ambiente externo ao computador.

Portas de comunicação de entrada e saída possuem características semelhantes as das classes ***geradores*** e ***indicadores***, respectivamente, considerando as simples funções de leitura e escrita nas devidas portas. Esta classe, porém, é mantida em separado, devido a previsão de outros dispositivos como uma porta de controle que não possui pinos de entrada ou saída, ou uma porta de protocolo possui pinos tanto de entradas como de saídas no mesmo dispositivo. Além disso, existe possibilidade de construir portas seriais, ou ainda, portas relacionadas com os sistemas contínuos, implementadas através de conversores A/D e/ou D/A, ampliando assim a classe comunicação.

A classe ***geradores*** compõem o sistema de alimentação ou excitação do circuito, composto por funções geradoras de sinais digitais ou analógicos. Os geradores digitais poderiam ser implementados por tabelas e pertencerem a classe *digitais*, porém, optou-se por mantê-los juntos em uma única classe e implementá-los através de algoritmos. Um fator que contribuiu para esta decisão é que o acesso à tabela se torna menos eficiente para o sistema, quando a relação número de entradas e possibilidades de valores de saída é muito pequena.

A classe ***indicadores***, contém dispositivos que servem para mostrar os resultados digitais e analógicos de um circuito.

5.1. DETALHAMENTO DAS CLASSES RELACIONADAS À PARTE ESTRUTURAL DO SISTEMA

O sistema como proposto na figura 29, é composto por três partes: controle geral do sistema; parte estrutural e interface com o usuário. As classes detalhadas a seguir não

"visíveis" ao usuário, visto que, tratam da estrutura do programa. Os atributos e métodos descritos nos próximos itens serão referentes a esta hierarquia.

5.1.1. A CLASSE ABSTRATA DISPOSITIVOS

Esta classe contém os atributos e métodos básicos que representarão dispositivos físicos e suas ligações no circuito.

Os atributos, neste caso, são o nome para o dispositivo, a definição do número de entradas e de saídas, a definição da operação por associação e finalmente as informações sobre as ligações com previsão de uma implementação por ponteiros que indicam quais os dispositivos ao qual o objeto corrente esta ligado.

Os principais métodos são: *ajustaEntrada*, que é responsável por atualizar as entradas com os valores fornecidos pelos objetos apontados; *ajustaSaida*, redefinida em cada classe, efetua a operação que define a função do objeto; *meLigaCom*, atualiza os campos dos ponteiros que efetivam a ligação entre objetos; *meDesligaDe*, faz a operação inversa, anulando o valor dos ponteiros; *verificaEntradas*, verifica se todas as entradas estão ligadas em alguma entrada de algum dispositivo.

O fato da classe ***dispositivos*** ser a classe base para todos as especializações que possuem diferentes funções que devem ser implementadas de forma virtual, e dos dispositivos serem combinados num único circuito, que na prática é uma lista, caracteriza o *polimorfismo*, visto que, a lista é do tipo ***dispositivos***, porém acomoda vários outros tipos.

Apesar de não ser obrigatório, mas devido a esta generalidade, a classe ***dispositivos*** é abstrata.

A maioria dos métodos desta classe, são referentes a manipulação exclusiva dos dispositivos, portanto devem ser conhecidos apenas pela própria classe pelas subclasses de nível imediatamente inferior, característica chamado no POO como informações protegidos.

5.1.2. A CLASSE DIGITAIS E CONTROLADAS

A classe ***digitais***, contém a descrição de *elementos mínimos* aplicados aos circuitos digitais que poderão ser aplicados complementarmente aos circuitos analógicos, porém, de uma forma homogênea no mesmo ambiente.

As portas lógicas, aparecem como os primeiros elementos desta classe, que deverá permitir a sua parametrização, para definir um objeto dinamicamente através de *atributos* e de *comportamento*.

A figura 31 mostra um exemplo de instanciação.

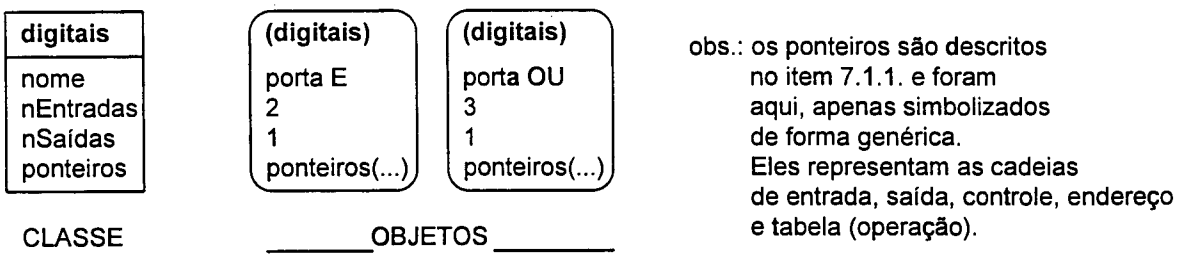


Figura 31. A classes ***digitais*** e possíveis objetos.

Como já foi dito a função de um circuito combinacional, será implementado pela sua tabela verdade. É previsto a geração de um arquivo contendo estas tabelas. Quando da instanciação de uma porta lógica ou outro dispositivo digital, é designado a este objeto, através de um mecanismo de associação, o ponteiro para tabela verdade que corresponde a função escolhida pelo usuário.

Pretende-se com isto, a criação de um mecanismo genérico de associação de classes, que possa abranger uma vasta gama de componentes com funções mínimas e macro-funções.

Como especialização desta classe, aparece a classe ***controladas*** para descrever os dispositivos de circuitos digitais que necessitam além das entradas de dados, entradas de

controle. E como a classe ***controladas*** herda as características de ***digitais***, ela utiliza o mesmo mecanismo de designação da função do dispositivo.

Para efetivar o controle do dispositivo, os métodos *meLigaCom* e *meDesligaDe* são redefinidos, e é criado o método *ajustaControle*, que atualiza as entradas de controle. O método *ajustaSaida* é também redefinido para atender os três tipos de controle discutidos no item 4.1.3, nível, borda e mestre-escravo. A escolha de um tipo de sincronização pode ser feita por parametrização.

Considera-se necessário, a utilização do mecanismo de tabelas apoiado por técnicas tradicionais de controle de fluxo de programa, para implementar o sincronismo com o relógio.

Com a junção destas duas metodologias, representar-se-á, dispositivos das áreas digitais, subáreas: circuitos combinacionais; seqüenciais; arquitetura de computadores e comunicação com o ambiente externo.

5.1.3. A CLASSE ANALÓGICOS

A classe ***analógicos***, contém a descrição dos *elementos mínimos* aplicados aos circuitos analógicos, a figura 32 mostra um exemplo de instanciação. Os elementos mínimos nesta área, são compostos por operações matemáticas básicas, ou por métodos numéricos como é o caso de um integrador, que poderá ser implementado por qualquer método de integração numérica.

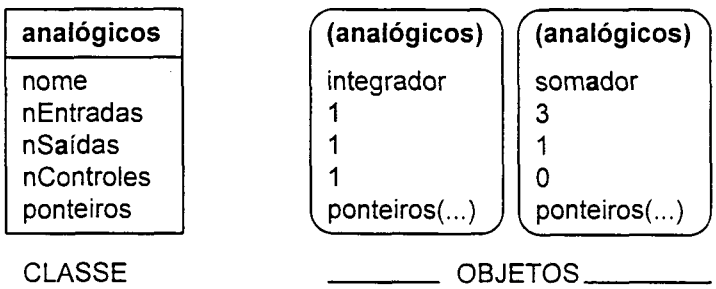


Figura 32. A classe analógicos e possíveis objetos.

Esta classe incrementa atributos para as entradas de controle, prevendo possibilidades de, por exemplo, reinicializar (zerar) o integrador, durante a simulação. Desta forma, é necessário redefinir alguns métodos como: *meLigaCom* e *meDesligaDe*, bem como, criar o método *ajustaControle*, da mesma forma descrita na classe ***controladas***. A função *ajustaSaída*, continua abstrata para que suas classes derivadas possam redefiní-las adequadamente.

A construção de tabelas, torna-se inviável neste caso, por se impossível prever valores obtidos, por exemplo, através de aquisição de dados em sistemas de tempo real. Considera-se, porém, para um segunda versão, a possibilidade de armazenar valores adquiridos em tabelas, para posterior utilização.

5.1.4. A CLASSE GERADORES

Denota-se de geradores, aqueles dispositivos que fornecem sinais para o sistema simulado, assim, devem ser ligados às entradas deste circuito. Possuem a característica especial de não apresentarem entradas para conexão (figura 33), visto que, sua função não depende de sinais externos, apenas de seus parâmetros. Para tal o atributo *nEntradas* deve ser inicializado com zero.

Para reger os geradores estabelece-se a classe abstrata ***geradores***, que fornece as características básicas para classes derivadas, que implementam diversos tipos de geradores digitais e analógicos.

Figura 33. Dispositivos geradores tem apenas saídas conectáveis.

É introduzido nestas classes, o método *alteraSaida*, que terá a função de incrementar ou decrementar o valor gerado na saída em tempo de simulação.

Estas classes implementam suas funções geradoras no próprio corpo da classe, e dentre os geradores digitais prevê-se:

• o gerador de corrente contínua, fornecendo como saída os valores lógicos constantes, **0** ou **1**, ajustados pelo usuário quando da sua criação;

• a chave liga/desliga, que poderá ser ligada ou desligada em tempo de simulação. Ligar ou desligar esta chave significa atribuir à sua saída, **1** ou **0** respectivamente à sua saída.

• a chave comutadora, composta com duas saídas que se alternam exclusivamente entre **0** e **1**. A mudança de valores é controlada pelo usuário em tempo de simulação.

• o gerador de onda quadrada, que alterna a geração de valores **0** e **1** em função de uma frequência pré-estabelecida pelo usuário em tempo de criação, com possibilidade de mudança desta frequência em tempo de simulação. Para controlar a geração desta freqüência torna-se necessário os atributos *periodo*, *frequencia*.

Dentre os geradores analógicos prevê-se:

• um segundo gerador de onda quadrada, que alterna a geração entre dois valores de amplitude regidos por uma freqüência. Estes valores são pré-estabelecidos pelo usuário em tempo de criação, com possibilidade de mudança em tempo de simulação.

• um gerador de constantes, onde o usuário fornece o valor da amplitude e valores limites, tanto superior como inferior, para dentro desta faixa, alterar o valor da saída em tempo de simulação;

• um gerador de onda senoidal, com possibilidades de freqüência e amplitude;

• um gerador da função exponencial;

- um gerador de onda dente de serra;

- um gerador de onda dente de serra;

- um gerador de pulso unitário (impulso);

5.1.5. A CLASSE INDICADORES

Denota-se de indicadores, aqueles dispositivos que recebem sinais do sistema simulado, assim, devem ser ligados às saídas deste circuito.

Por assim serem, possuem a característica especial de não apresentar saídas com possibilidades de conexões (figura 34). Assim, o atributo *nSaidas* deve ser inicializado com zero.

Figura 34. Dispositivos indicadores tem apenas entradas conectáveis.

Para reger os indicadores estabelece-se a classe abstrata ***indicadores***, que fornece as características básicas para classes derivadas que implementam diversos tipos de analisadores de sinais digitais e analógicos, que são implementados por funções, principalmente gráficas, inerentes a linguagem de programação, dentre eles prevê-se:

- **led** que pode estar aceso ou apagado. Esta mudança ocorre de acordo com a saída do dispositivo ao qual está ligado. Graficamente, mostrar-se-á o desenho de um disco colorido de branco para representá-lo apagado, e de vermelho para representá-lo aceso, isto ainda corresponde respectivamente os valores lógicos **0** ou **1** recebido em sua entrada;

- um **display** de sete segmentos mostra os valores binários recebidos em suas entradas convertidos para a forma de números hexadecimais, portanto, faz-se necessário um atributo (*valor*) para armazenar o valor convertido;

• através de um *analisador lógico*, poderá visualizar-se o diagrama de tempo referente aos valores de saída de diversas saídas de dispositivos ou circuitos digitais. As saídas são escolhidas através de pontas de prova, que são a caixa do dispositivo em si, porém, para desenhar o diagrama de tempo necessita-se de mais espaço, desta forma prevê-se a alocação, no momento da instalação, de uma "janela" para cada analisador. Para coordenar estes desenhos, cria-se os seguintes atributos: o ponteiro **janela*, um contador (*nAnalisador*) para saber o número de janelas alocadas e a partir daí definir o local da instalação da janela;

• um *osciloscópio* mostra os gráficos de funções contínuas na forma bidimensional, é utilizado para acompanhar a evolução de dispositivos ou circuitos analógicos, ligados também através de pontas de prova. O atributo *corTraco* define uma cor de traço para cada ponta de prova.

5.1.6. A CLASSE COMUNICAÇÃO

Através de dispositivos de comunicação, pode-se alimentar um circuito simulado, com informações oriundas de componentes físicos reais, e ainda, enviar resultados obtidos pela simulação, para os mesmo circuitos externos ao computador.

Estas ações, são comparáveis à dos geradores e indicadores, porém, efetuadas fisicamente através de portas de comunicação do computador, e para tanto, faz parte deste trabalho o projeto e a construção de uma interface de **hardware** (já concluídos), contendo portas de comunicação de entrada, saída, controle e protocolo. Esta placa já esta construída e é apresentada no apêndice C.

A classe ***comunicação***, descreve dispositivos que realizam a comunicação com o ambiente externo, e deve apresentar subclasses relacionadas com cada operação possível com a placa de interface. Além da simples leitura e escrita em portas físicas, poder-se-á, programar a interface e manter um protocolo de comunicação entre o simulador e o ambiente controlado.

A classe ***portaEntrada*** descreve a porta pela qual é feita a leitura dos dados externos. Ela é caracterizada por uma função que lê dados em um endereço no espaço de E/S do computador e os deixa disponível em suas saídas. O endereço pode ser determinado pelo usuário na criação do objeto. Para coordenar estas ações são redefinidos métodos *ajustaEntrada* e *ajustaSaida*.

A classe ***portaSaida***, descreve a porta pela qual é feita a escrita em ambientes externos. Ela escreve em um endereço definido pelo usuário, dentro do espaço de E/S do computador. Apenas o método *ajustaSaida* é redefinido.

As classe ***portaControle*** e ***portaProtocolo***, são previstas para comunicar com dispositivos reais de entrada e saída que são programáveis e que necessitam de mecanismos de sincronização, como uma PPI programável.

O projeto da interface **hardware** e detalhes sobre sua programação, protocolo de comunicação e outros pontos relacionados, são abordados oportunamente no apêndice B.

5.2. A CLASSE CIRCUITO

A classe ***Dispositivos*** e suas derivadas, são as classes "visíveis" para o usuário do simulador, porém, para possibilitar a sua funcionalidade, são utilizadas várias classes responsáveis, por exemplo, pela interligação e comunicação entre os dispositivos. Estas são descritas a seguir.

Todos os componentes de um sistema são imprescindíveis para a sua completa realização, porém, a classe ***Circuito*** recebe um destaque entre as outras, por ser a classe que manipula ou acessa todas as outras até agora descritas.

Ela é composta por vários métodos responsáveis principalmente pela:

- instalação e remoção de dispositivos (*insere* e *remove*);

- coordenação e verificação das ligações entre os dispositivos (*liga*, *desliga* e *verificaLista*);
- otimização, depuração e simulação do circuito (*otimizaLista*, *depuraLista* e *processaLista*).

O processo usual para estes tipos de tarefas, é a utilização de *listas*. Prevê-se para esta classe, que os métodos citados sejam efetuados sobre uma lista genérica de ***dispositivos***.

Efetivamente esta é uma lista de objetos, portanto, os objetos a serem inseridos apresentam o tipo *dispositivos*. Este fato denota a aplicação do conceito de polimorfismo sobre esta lista, qual pode-se denominar de lista polimorfa.

A simulação do circuito significa o percorrimento iterativo desta lista, atualizando as entradas e saídas dos dispositivos, por tempo indeterminado. A depuração, significa o percorrimento da lista apenas uma vez.

5.3. DETALHAMENTO DAS CLASSES RELACIONADAS À INTERFACE COM O USUÁRIO

Nos itens anteriores estudou-se as classes que tratam da estrutura e da funcionalidade do sistema a ser simulado. Estas classes, são apoiadas pelas classes ***gráfica***, ***geral*** e ***menu*** que permitem a construção de sistemas de forma gráfica e iterativa. Através delas o usuário mantém contato com o simulador.

5.3.1. A CLASSE GRÁFICA

A partir da seqüência de atributos desta classe apresentados na figura 35, *matriz* forma a base da interface com o usuário.

A tela do monitor de vídeo é dividida na forma de uma matriz virtual por onde caminha um cursor visível. Os quatro seguintes atributos são referentes a esta matriz, tratando do posicionamento do cursor e conseqüentemente dos dispositivos. Em seguida vem os atributos referentes ao desenho dos dispositivos e das ligações.

A seguir apresenta-se as principais atividades da classe ***grafica***, bem como, os métodos envolvidos:

- o desenho e a movimentação do cursor na tela (*desenha cursor*, *apagaCursor* e *movimentaCursor*);
- o desenho e remoção dos dispositivos no vídeo, enumerando-os para futura identificação (*desenhaComponente*, *apagaComponente*);
- a interligação e desligamento visual, entre a entrada de um dispositivo à saída de outro (*desenhaLigação* e *apagaLigação*);
- o armazenamento e a carga em disco de um circuito construído (*salvaCircuito* e *carregaCircuito*).

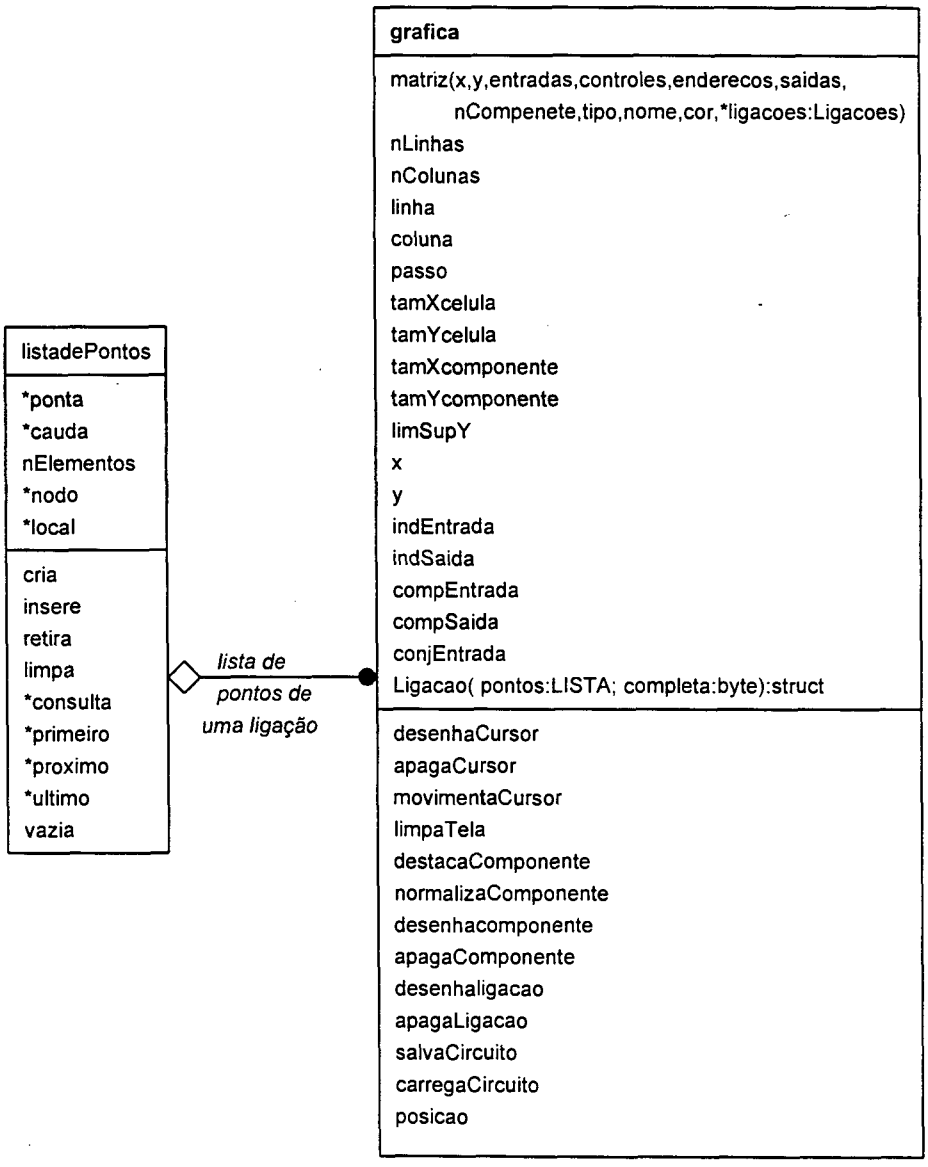


Figura 35. A classe ***gráfica***, e sua associação com a lista de pontos de uma ligação.

Além destas atividades, a classe ***grafica*** passa mensagens para a classe ***circuito***, para que esta possa efetuar as tarefas estruturais paralelamente a construção do circuito pelo usuário.

Para manter a seqüência de pontos da tela, utilizados para desenhar a linha que liga um dispositivo ao outro, prevê-se uma lista de pontos, referentes as coordenadas do vídeo.

Esta lista se faz necessária, devido há possíveis correções nesta ligação, em tempo de construção do circuito, e também são informações extremamente úteis na reprodução deste circuito quando carregado do disco.

5.2.2. A CLASSE MENU

A classe ***Menu*** (figura 37) fornece suporte para a comunicação do usuário com o programa. Apresenta menus janelados tipo popdown, que permitem o usuário disparar as tarefas contidas nos métodos das classes ***Circuito*** e ***Gráfica***.

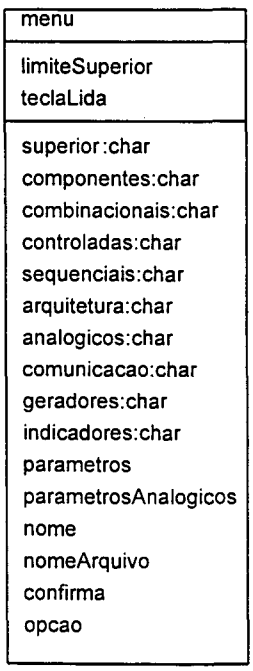


Figura 37. A classe ***menu***.

5.2.3. A CLASSE GERAL

A classe ***geral*** (figura 38) fornece as funções relacionadas a entrada e saída de dados, através dos dispositivos padrão do computador: o teclado como entrada o monitor de vídeo como saída.

As principais atividades e métodos são:

- desenhar e movimentar os menus popdown (*superior*);
- oferecer menu para escolha do componente ou dispositivo (*componentes, combinacionais, controladas, sequenciais, arquitetura, analogicos, comunicacao, geradores, indicadores*);
- oferecer menus para parametrização dos dispositivos (*parametros* e *parametrosAnalogicos*).

Os atributos se resumem em um *buffer* para armazenar as mensagens a serem enviadas, e a cor que deve variar entre a mensagem de estado do simulador para mensagens de erro.

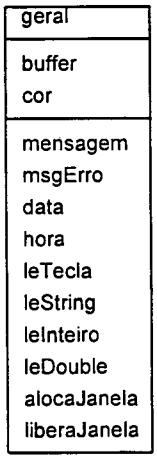


Figura 38. A classe ***geral***.

As principais atividades de entrada e seus respectivos métodos são:

- leitura de uma tecla, um sentença de caracteres, de um número inteiro ou um número real (*leTecla*, *leString*, *leInteiro*, *leDouble*).

Como atividades de saída, destaca-se:

- construção e remoção de janelas;
- envia mensagem de estado e erro (*mensagem* e *msgErro*).

Outras funções com conotação ilustrativa:

- mostra a data e a hora do sistema (*data* e *hora*).

Esta classe, pela sua utilidade, deve ser composta essencialmente por métodos públicos que podem ser acessados, através de mensagens, por todas as outras classes do sistema.

5.3. A TROCA DE MENSAGENS ENTRE AS CLASSES

A notação utilizada, é sugerida por Embley [18], onde, os quadros representam as classes e traços com setas no final representam respectivamente, as mensagens e o sentido do envio. A tabela 5 mostra um resumo da notação utilizada.

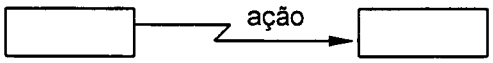
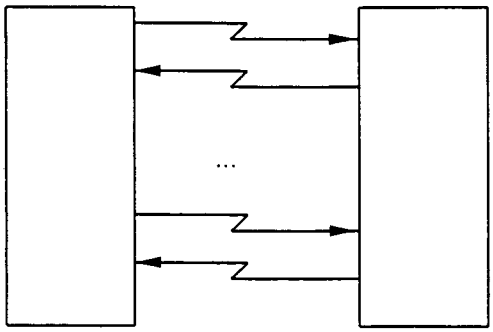
INTERAÇÃO ENTRE OBJETOS	SÍMBOLO
simples	
seqüência interativa	

Tabela 5. Notação utilizada para representar a troca de mensagens entre as classes.

Ressalta-se que nem todas as linguagens de programação, implementam o conceito de superclasse como SmallTalk. Em C++, por exemplo, a técnica utilizada é do programa principal (**main**) controlando o fluxo do sistema. Baseando-se nesta técnica passa-se a estudar alguns casos de troca de mensagens entre as classes.

5.3.1. EXEMPLO DE INSTALAÇÃO DE UM DISPOSITIVO

O usuário decide através das quatro setas do teclado, a posição onde quer instalar o dispositivo.

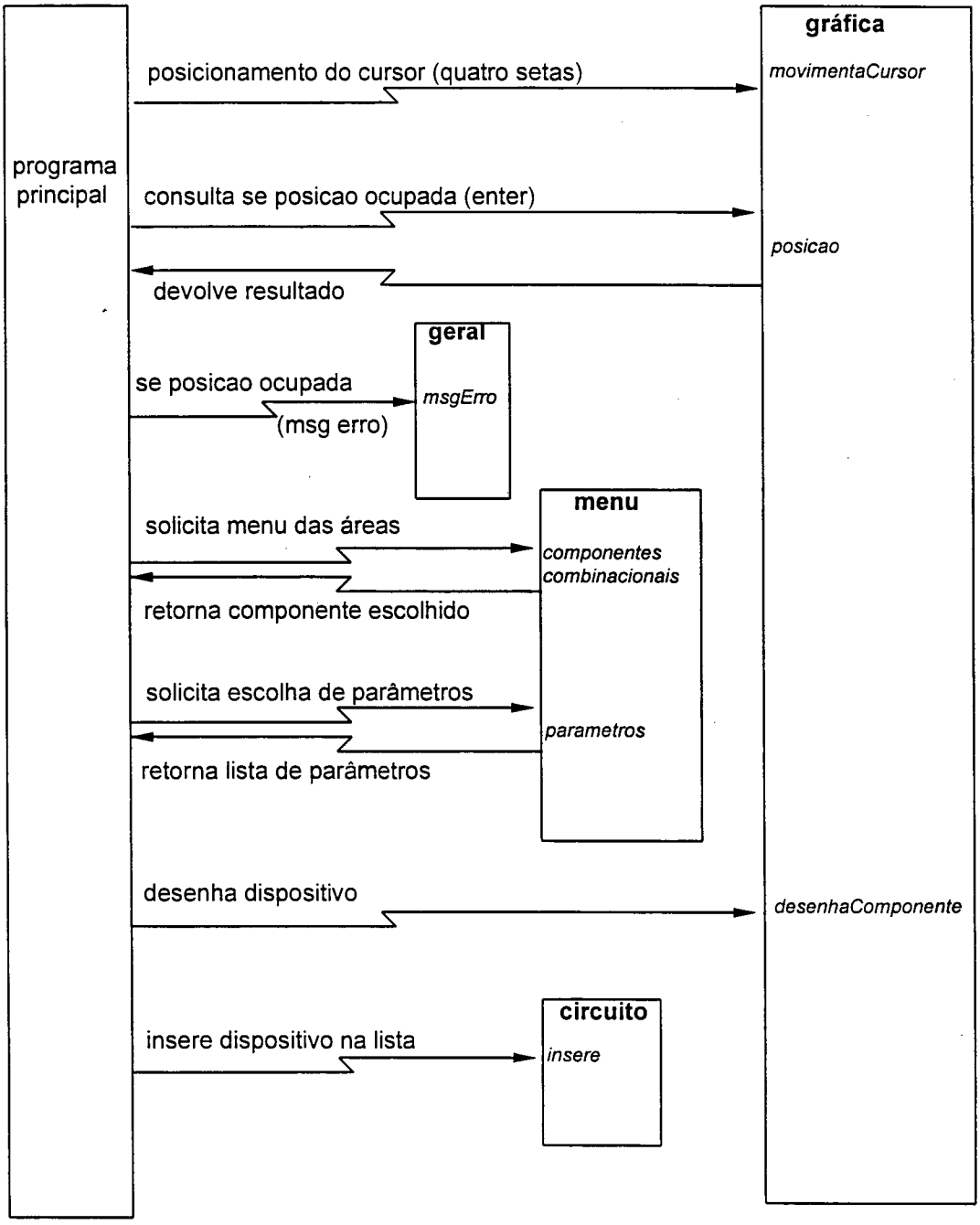


Figura 39 Sincronização das classes quando da instalação de um dispositivo.

Após isto, ao digitar-se a tecla **enter**, é verificado se a instalação não está sendo feita sobre um outro dispositivo. Caso positivo, uma mensagem é enviada para o vídeo avisando o usuário, caso contrário, aparece um menu onde o usuário pode parametrizar o dispositivo em instalação, e finalmente, ele é desenhado no vídeo, apresentando a entradas à esquerda do dispositivo e as saídas à direita.

Durante a instalação e de forma transparente para o usuário, este dispositivo é inserido na lista polimorfa de dispositivos.

A figura 39 mostra a seqüência que é necessária para instalar um dispositivo.

5.3.2. EXEMPLO DE INTERLIGAÇÃO ENTRE DISPOSITIVOS

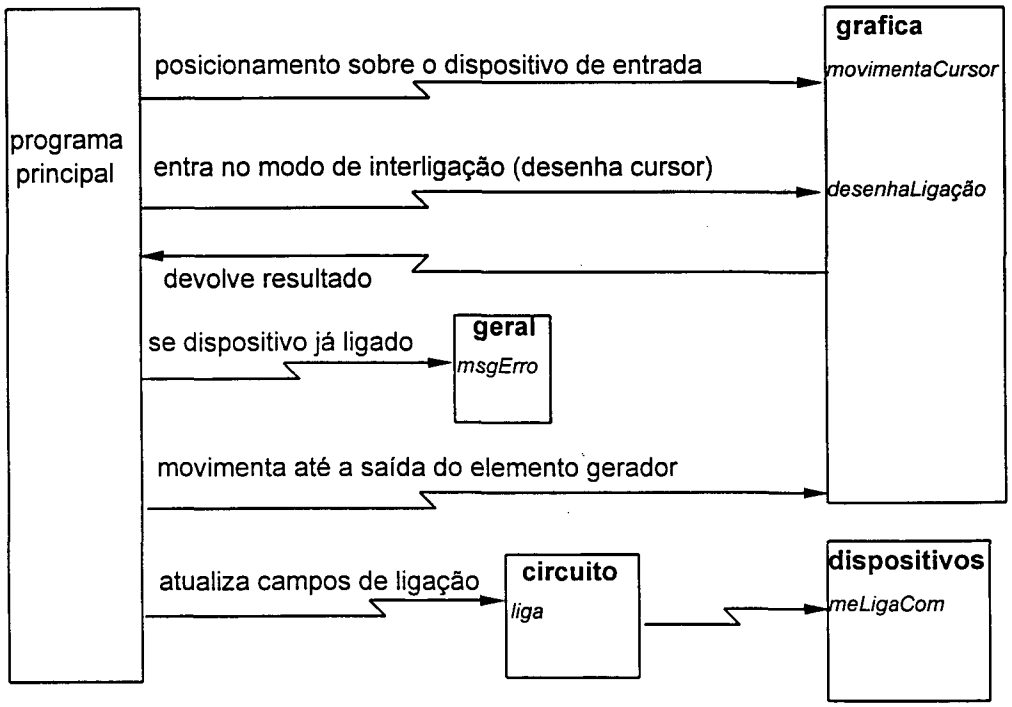


Figura 40. Exemplo da interligação entre dois dispositivos.

Na interligação, é exigido devido a estrutura adotada, que o usuário posicione o cursor primeiramente sobre as entradas do dispositivo que receberá os dados, e depois sobre as

saídas do dispositivo que fornecerá os dados. Os campos tipo ponteiro são atualizados efetivando a ligação.

5.3.3. EXEMPLO DE SIMULAÇÃO DO CIRCUITO

A simulação envolve o percorrimento da lista, que é pesquisada somente se todos os dispositivos tem suas entradas completamente ligadas, veja ilustração na figura 41.

Além disso, está previsto uma otimização no percorrimento desta lista, e sempre que uma lista for criada ou um circuito modificado, esta otimização é efetuada.

O processamento da lista significa o seu percorrimento cíclico, ajustando as entradas e ajustando as saídas dos dispositivos, até que haja uma intervenção do usuário.

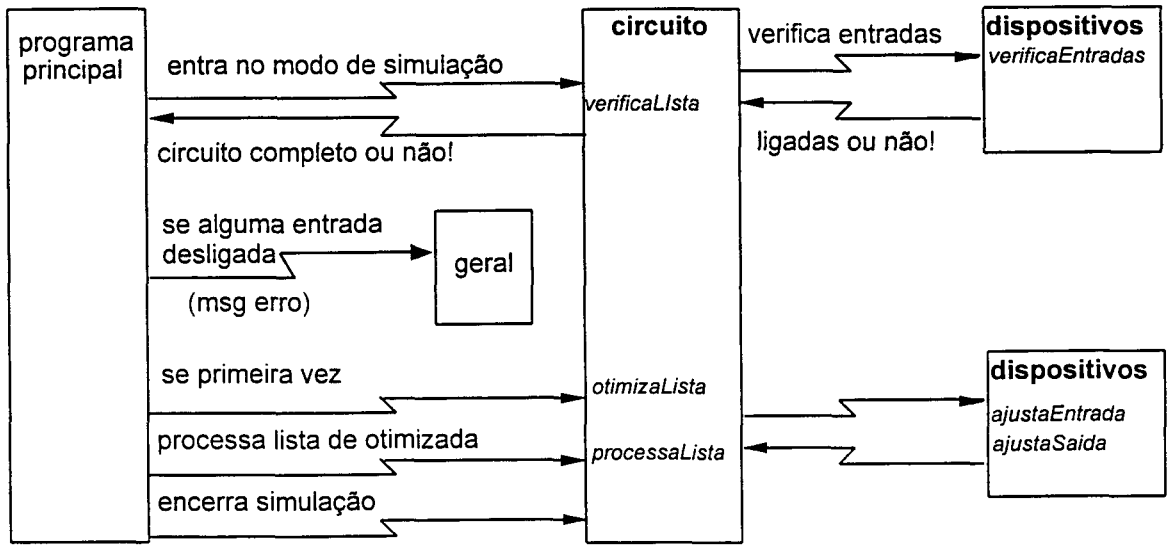


Figura 41. Exemplo de simulação.

6. IMPLEMENTAÇÃO DO SIMULADOR *lab*GRAF

Os trabalhos desenvolvidos durante a elaboração desta dissertação resultou na implementação de um sistema computacional que busca atender as necessidades apontadas durante seu decorrer. Este sistema, alusivamente à um laboratório de eletrônica, implementado em um ambiente gráfico, é denominado de ***lab***GRAF, ou seja, *laboratório gráfico*.

Este sistema foi escrito na linguagem orientada a objetos C++ e compilado pelo **Microsoft® C/C++ Compiler** versão 7.00 [63].

Este sistema é composto por um conjunto de arquivos, onde cada um deles, corresponde diretamente à uma classe, ou à um conjunto de classes com características em comum, de acordo com a tabela 5.

arquivo de cabeçalho	arquivo corpo das classes	aplicação (tipo de classes)
dispositivos.h	dispositivos.cpp	dispositivos e especializações
circuito.h	circuito.cpp	lista de dispositivos
folha.h, lista.h	folha.cpp, liga.cpp, lista.cpp	ambiente de trabalho
menu.h	menu.cpp	menus do usuário
geral.h	geral.cpp	mensagens e outros
	lagraf.cpp	programa principal

Tabela 5. Nomes dos arquivos com código fonte.

Os arquivos de cabeçalho (**header**), contém os protótipos das classes, os arquivos (cpp) contém a implementação em si, com seus atributos, métodos e demais características.

A instanciação das classes ***grafica***, ***menu*** e ***geral*** é feita automaticamente de forma embutida nas próprias classes (`Grafica grafica; Menu menu; Geral geral;`), onde, o nome com letra maiúscula corresponde à classe (ou ao tipo), e o nome com letra minúscula ao objeto. Desta forma, o programa principal já conta com os métodos públicos destas classes à sua disposição.

A manipulação de todos as classes é feita através do arquivo "labgraf.cpp", ou seja, o programa principal, que tem como primeira tarefa, instanciar a classe ***circuito*** (`Circuito = new circuito`), criando assim a lista onde os dispositivos são instalados (veja detalhes desta lista no item 6.3.1.).

Após isto, o programa principal entra num laço do tipo `do{...}while;` encerrado pelo usuário através de uma tecla de controle, como mostra o trecho de código da listagem 2.

```
//usuário solicita finalização através da tecla alt_x ou alt_F4
case FIM: if(!salvo)    //caso o circuito foi modificado deve ser salvo
          if (menu.confirma("Circuito modificado, salvar? S/N") == OK){
             if (strcmp(nomeArq,"semNome")==0) //obtem um nome para o
circuito
                menu.leNomeArquivo(nomeArq,"Salva");
             if (folha.salva(nomeArq) != ERRO) //previne erro no salvamento
                salvo = 1;
          }
          if (menu.confirma("Abandona o labGRAF? S/N") == OK){
             delete circuito;       //encerra lista de dispositivos
(circuito)
             return OK;             //agora sai
          }
          break;
```

Listagem 2. Trecho do programa principal que dá saída ao DOS.

Dentro deste laço acontece todo o fluxo do sistema. Através de teclas de controle, disponíveis para o usuário (veja manual do usuário no apêndice A), ele controla a instalação, remoção e interligação dos dispositivos (o que corresponde à construção de um circuito), carga, salvamento e simulação de um circuito, acesso à tela de ajuda e finalização do ***lab***GRAF.

Os demais arquivos estão detalhados a seguir. Além disso, uma listagem completa do sistema é fornecida, que pelo grande volume apresentado, não é anexada neste trabalho, porém, está à disposição na coordenadoria do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina.

6.1. A IMPLEMENTAÇÃO DA ESTRUTURA DO DISPOSITIVO

No projeto deste sistema foi visto que, ***dispositivos*** é classe mais geral dos componentes e contém atributos que são compartilhados por todas as subclasses.

A figura 42, mostra uma estrutura genérica contendo a função abstrata pura (2), o que torna também, a própria classe abstrata.

A entrada (1), é uma cadeia de valores, que recebe dados de elementos geradores[4], e a saída (3), outra cadeia do mesmo tipo que recebe os valores tratados ou modificados pela função (2) do dispositivo.

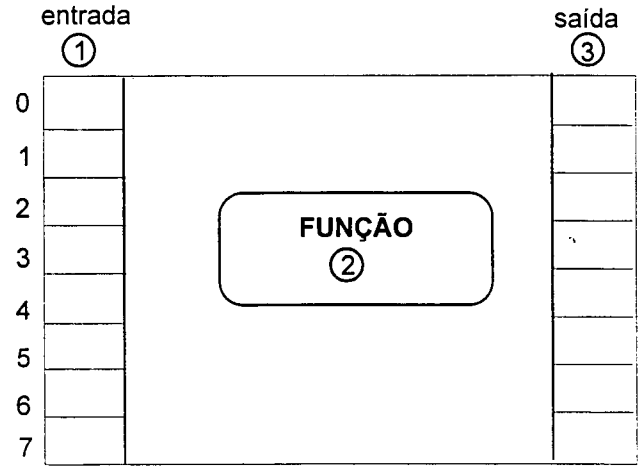


Figura 42. Estrutura básica dos dispositivos, contida na classe mais geral do sistema.

Para tornar os elementos desta estrutura básica mais genérica, estas cadeias devem ser dinâmicas, ou seja, o seu tamanho deve ser definido no momento da instanciação do objeto. Para tal, é necessário campos para parametrizar esses objetos. A figura 43, apresenta os atributos "tamanho da cadeia de entrada" (2) e "tamanho da cadeia de saída" (3), utilizados para esta finalidade.

---

[4]Entenda-se por elementos geradores, não somente as entradas geradoras, mas qualquer objeto que forneça dados para outro.

A estrutura de um sistema é representado pelas ligações entre seus elementos. Ainda na figura 43, pode-se observar campos (5), através dos quais serão efetuadas estas ligações estruturais.

Como cada entrada poderá estar ligada à um elemento diferente, há a necessidade de uma cadeia (5) do mesmo tamanho de *entrada* (1), para alojar as informações dos objetos geradores, utilizadas para a interligação.

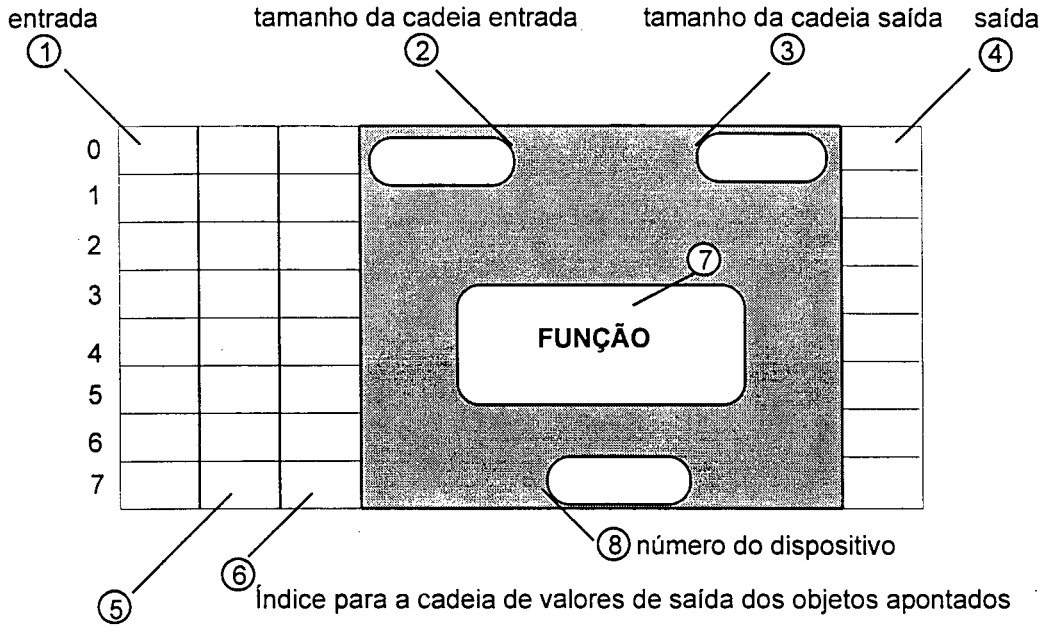


Figura 43. Estrutura básica da classe ***dispositivos***.

Estas informações são endereços, referentes ao objeto gerador está fixado durante a execução do programa. Estes endereços são implementados por *ponteiros*.

Um objeto pode estar apontando para vários outros objetos, proporcionalmente ao número de entradas que ele possui.

Além disso, o objeto apontado pode conter uma cadeia de valores de saída com mais de uma posição, por exemplo, um decodificador 3x8, que possui oito saídas.

Necessariamente, para resolver este problema, deve-se ter um atributo para indicar qual posição da cadeia de saída do objeto apontado, dever ser copiada para a determinada entrada do objeto receptor ou apontador, o atributo (6) é o índice para a cadeia de valores de saída dos objetos apontados.

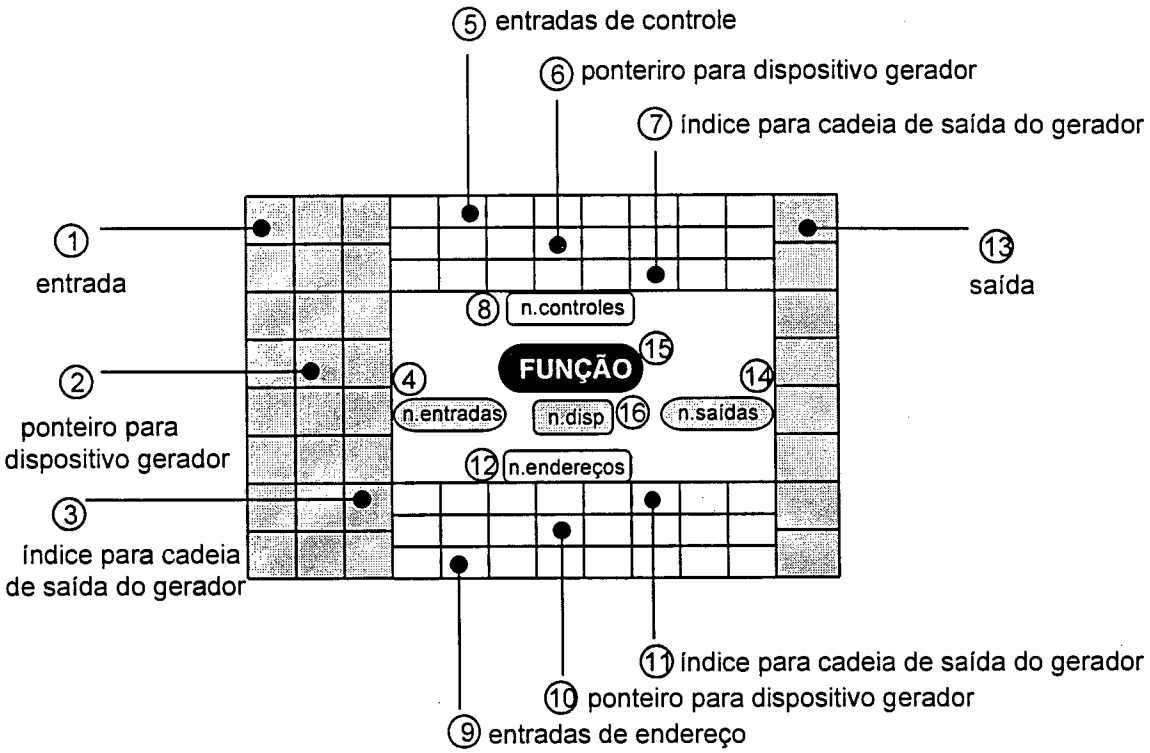


Figura 44. Uma especialização da classe ***dispositivos***.

Além destes atributos, é mantido um campo para armazenar o número do dispositivo (8), quando da sua instalação. Este número esta relacionado com sua posição no ambiente de trabalho que será discutido com detalhes na interface gráfica.

Tendo em vista que a classe ***dispositivos*** conterá as características básicas, as subclasses específicas, deverão incrementar as características necessárias para descrever dispositivos específicos. Por exemplo, na figura 44, é ilustrada uma especialização por herança da estrutura acima. Trata-se da inclusão dos campos para implementar a classe ***controlados***.

Os campos *controle* (4) e *endereço* (7), visam suprir as necessidades de dispositivos físicos que possuem terminais de habilitação e/ou endereço respectivamente. Além disso, como todas as cadeias de valores devem ser dinâmicas, os campos *número de controles* (6) e *número de endereços* (9) servem para definir o tamanho respectivo de cada cadeia, tal qual, os campos *número de entradas* (3) e *número de saídas* (11);

6.1.1. DESCRIÇÃO DOS ATRIBUTOS DA ESTRUTURA BÁSICA

Aqui são descritos os tipos de dados e aplicações dos atributos, ressalta-se que todas as cadeias são limitadas à oitos posições como discutido no final deste item.

Passa-se a descrever atributos que considera-se, essenciais para compreensão deste trabalho.

a) número do componente

- Tipo: **byte**.
- Conteúdo: o número utilizado para identificação do dispositivo.

b) número de entradas

- Tipo: **byte**.
- Conteúdo: o tamanho da cadeia de valores de entrada.

c) entrada

- Tipo: união entre uma cadeia de **bytes** e uma cadeia de **double**. Isto é feita para representar dispositivos digitais e analógicos, respectivamente.
- Conteúdo: cada posição deste vetor, corresponde a uma entrada do dispositivo corrente. Os valores contidos nele, são obtidos através da simples cópia da saída de outro dispositivo, apontado pelo respectivo ponteiro (item d).

d) ponteiro para saída do objeto gerador

- Tipo: cadeia de ponteiros.
- Conteúdo: ponteiros para os objetos que simulam dispositivos

e) índice para a cadeia de valores da saída

- Tipo: cadeia de **bytes**.
- Conteúdo: cada posição desta cadeia, armazena um índice para o vetor de saída do dispositivo ao qual aquela entrada esta ligada, este atributo é necessário porque determinados dispositivos podem ter mais de uma saída.

f) numero de saídas

- Tipo: **byte**.
- Conteúdo: o tamanho do vetor de valores de saída.

g) saída

- Tipo: união entre uma cadeia de **bytes** e uma cadeia de **double**.
- Conteúdo: cada posição deste vetor, corresponde a um valor da saída do dispositivo corrente, seja digital ou analógico. Os valores contidos nele são obtidos pela função do dispositivo, que operou sobre os valores da entrada.

Neste sistema para compor o tamanho máximo das cadeias de entrada, saída, controle e endereços, considera-se oito um número adequado, pois, em geral este tipo de componentes não excede a este limite, e modelos matemáticos de dispositivos analógicos, em geral não atingem este limite;

Este número também permite que a memória exigida para armazenar um circuito, não tenha um tamanho exorbitante. Por exemplo, se cada posição das cadeias de um dispositivo digital ocupa um **byte**[5], seriam necessários, oito **bytes** para representar a entrada (1), oito para o índice (3) para a cadeia de saída do elemento que gera sinal para estas entradas e oito para a saída (13), totalizando vinte e quatro **bytes**.

Os ponteiros são formados (num computador que utiliza um sistema operacional do tipo DOS) por dois **bytes** que apontam para o segmento de memória e dois para o

---

[5] Em C/C++, o tipo **byte** é definido pela declaração **unsigned char**, e suporta valores de 0 a 255.

deslocamento dentro deste segmento, perfazendo um total de trinta e dois **bytes** para uma cadeia de oito ponteiros.

Num componente que possua oito entradas e oito saídas, apenas para representar o grupo de cadeia de entrada e a cadeia de saída, seriam necessários cinqüenta e seis **bytes**. Já num dispositivo da classe ***controlados***, seriam necessários, além destes, mais noventa e seis **bytes** para representar seus atributos das entradas de controle e entradas de endereço.

Supondo ainda um circuito com cinqüenta e seis componentes, todos com as cadeias de oito posições ocupadas, obteria-se um total de 8.512 **bytes**, ou seja, pouco mais de 8 **Kbytes**.

Os métodos implementados ou ponteiros para os métodos e funções ocupariam mais algumas posições de memória, quantidade tal, que varia de um componente para outro proporcionalmente à implementação de seus métodos.

No caso dos dispositivos analógicos, que exigem mais espaço para representar seus valores, o tamanho da cadeia de entrada e saída ficam reduzidos em geral para uma posição cada. O tipo utilizado nesta implementação é o **double**[6] do compilador C/C++, que é utilizado para representar os números reais, e ocupa oito **bytes** na memória.

Esta rápida análise, não inclui ainda, atributos auxiliares, atributos específicos das subclasses, informações necessárias para a interface gráfica (como as listas de ligações), e outros atributos considerados detalhes de implementação neste escopo. Porém, pode-se ver que não existe, a princípio, uma sobrecarga de utilização de memória.

Além disso, dado o dinamismo dos objetos, nem todos os dispositivos, num mesmo circuito, terão todas as possíveis entradas e saídas, portanto, serão criados objetos com

---

[6] **Double** é a representação de valores em ponto flutuante pela linguagem C/C++. A faixa de valores permitidos por este tipo é de $.7 \times 10^{-308}$ a $.7 \times 10^{308}$.

números menores que oito entradas ou saídas, e neste caso o número de posições de memória seria reduzido proporcionalmente.

Concluindo esta análise, visualiza-se possibilidades de expansão destes números.

6.2. AS FUNÇÕES DOS DISPOSITIVOS

A hierarquia das classes foi montada basicamente de acordo com as áreas digital e analógica. Os próximos itens descrevem as classes e os objetos relacionadas com cada área de aplicação, conforme seguido nos capítulos e itens anteriores.

Para os dispositivos digitais, optou-se por uma implementação baseada em tabelas. Esta decisão foi tomada visando viabilizar também funções complexas que representam, por exemplo *circuitos integrados de lógica programável*, *máquinas de estado*, bem como a definição por parte do usuário de dispositivos com características que servem especificamente à sua aplicação em desenvolvimento.

Esta última característica, representa um grande avanço em relação aos simuladores estudados, que na sua grande maioria, não permitem a definição de novos dispositivos.

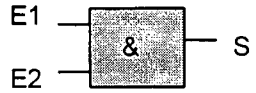
6.2.1. CIRCUITOS DIGITAIS COMBINACIONAIS

Os valores contidos na cadeia de entrada de um dispositivo, são utilizados para formar o índice para uma tabela que contém os possíveis valores de saída do dispositivo.

Como foi discutido no capítulo 4, a lógica binária exige, como seu nome diz, a presença de apenas dois valores, porém, circuitos eletrônicos reais, são compostos também por dispositivos com saídas no *terceiro estado* e, eventualmente, dispositivos com entradas sem valores definido e outros.

Considerando esta possibilidade, as tabelas foram construídas levando-se em conta o valor do *terceiro estado*, e adotou-se para representá-lo o número dois (**2**). Desta forma, matematicamente têm-se um sistema ternário, ao invés de um sistema binário.

Na figura 45, pode-se examinar o exemplo de uma porta **E**, representada através de sua tabela verdade composta pelos valores **0**, **1** e **2**.



ind	E2	E1	S
0	0	0	0
1	0	1	0
2	0	2	0
3	1	0	0
4	1	1	1
5	1	2	2
6	2	0	0
7	2	1	2
8	2	2	2

Figura 45. Uma porta **E** com duas entradas E1 e E2 e uma saída S.

O acesso à esta tabela é ilustrado no trecho de código da listagem 3.

```
unsigned char indice = 0;                                  //índice para a tabela
for (unsigned char i=0; i<nEntradas; i++)
    indice += entrada[i] * (unsigned char) pow(3,i); //cálculo do índice
saida[indSaida] = tabela[indice];                          //acessso à tabela
```

Listagem 3. Acesso à tabela ternária.

Os circuitos digitais combinacionais são supridos pelas classes ***digitais*** e ***controlados***.

A classe ***digitais*** pode ser instanciada pelas seguintes portas lógicas: *e*, *ou*, *não e*, *não ou*, *ou exclusivo* e *inversor*, construídas através de tabelas como as da figura 45, e tem como símbolos básicos, uma caixa contendo no seu interior a indicação da função, conforme mostra a tabela 3.

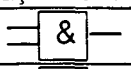
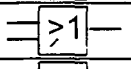
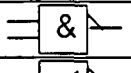
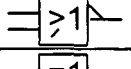
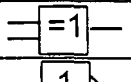
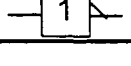
função lógica	indicação da função
e	
ou	
não e	
não ou	
ou exclusivo	
inversor	

Tabela 3. Símbolos para as funções lógicas.

Com exceção do *inversor*, todas as portas lógicas foram implementadas com duas, três e quatro entradas.

A classe ***controlados*** será instanciada pelas mesmas portas lógicas descrita acima, adicionadas de um terminal de controle de habilitação, e por macro-funções relacionadas à área, que podem conter além de terminais de controle, terminais de endereçamento. Portas lógicas com terminais de controle não são encontradas em componentes comerciais, porém, resolveu-se criá-las aproveitando a estrutura criada, dado o baixo custo de implementação

O acesso às tabelas das portas lógicas, dar-se-á, então somente após a confirmação da validação das entradas de controle do componente, que pode se dar por habilitação nível como mostra a listagem 4 (mais utilizados nos circuitos assíncronos) e pela transição do relógio (aplicado nos circuitos síncronos).

```
for (i=0; i<8; i++)
    saida[i] = 2;                          //pré-define como terceiro estado
for (i=0; i<nControles; i++){
    ajustaControle(i);
    if (controle[i] == DESABILITADO){
       return OK;                          //saída permanece em terceiro
estado
    }
}
unsigned char indice = 0;
for (unsigned char i=0; i<nEntradas; i++)
    indice += entrada[i] * (unsigned char) pow(3,i);
saida[indSaida] = tabela[indice];          //saída recebe valor binário
return OK;
```

Listagem 4. Acesso à tabela somente após a habilitação.

Dentre as macro-funções possíveis, este sistema oferece um *decodificador 3x8* com um terminal de habilitação, e um *multiplexador 4x1* com dois terminais de endereço. Os símbolos adotados para estes dispositivos, forma mostrados na figura 18 do item 4.3.1.

Pelo fato do decodificador possuir mais de uma saída (oito), existe uma diferença entre sua tabelas e as estudadas até agora. Isto acarreta num algorítmo exclusivo para atualizar estas saídas, desta forma, torna-se necessário a criação de uma subclasse de ***controlados*** exclusiva para representá-lo. A listagem 5 mostra a implementação deste algoritmo.

```
for (i=0; i<8; i++)
    saida[i] = 0;              //prepara saídas: todas desativadas
for (i=0; i<nControles; i++){
    ajustaControle(i);
    if (controle[i] == DESABILITADO){
       return OK;              //mantém todas as saidas com zero
    }
}
unsigned char indice = 0;
for (unsigned char i=0; i<nEntradas; i++)
    indice += entrada[i] * (unsigned char) pow(3,i);
indice = tabela[indice];       //tabela contém número da saída a ser ativada
saida[--indice] = 1;           //decodificação em si, ativa saída com 1
return OK;
```

Listagem 5. Trecho de código que implementa o decodificador 3x8.

O multiplexador, como mostrado na listagem 6, é implementado somente com a utilização de tabela, porém como possui terminais para endereçamento, estes devem entrar no cálculo do índice da tabela, esta divergência cria a necessidade de manter uma subclasse de ***controlados*** para representá-lo.

```
unsigned char indice = 0;
// avalia os valores na entrada de dados
for (unsigned char i=0; i<nEntradas; i++)
    indice += entrada[i] * (unsigned char) pow(3,i);
// avalia os valores na entrada de controle somando ao índice anterior
for (; i<nControles+nEntradas; i++)
    indice += controle[i-nEntradas] * (unsigned char) pow(3,i);
saida[0] = entrada[indice];          //acesso à tabela
```

Listagem 6. Trecho do código que implementa o multiplexador.

6.2.2. CIRCUITOS DIGITAIS SEQÜENCIAIS

Esta área utiliza as mesmos classes citadas no item anterior e incrementa alguns objetos instanciados da classe ***controlados***, como o **latch** e **flip-flip** tipo D e algumas subclasses como o **flip-flip** tipo JK e a máquina de estados.

Os **flip-flops** as máquinas de estado possuem um sincronismo com o relógio diferente dos **latchs**. Os algoritmos utilizados para implementar a sincronização são mostrados na listagem 7, pelo nível (parte a), pela transição positiva (parte b) e transição negativa (parte c).

```
//parte a: nível alto
if (controle[0] == 1)

//parte b: transição positiva ou borda de subida
if ((controleAnterior == 0) && (controle[0] == 1))

// parte c: transição negativa ou borda de descida
if ((controleAnterior == 1) && (controle[0] == 0))
```

Listagem 7. Tipos de controle dos dispositivos.

Optou-se por implementar os **flip-flops** JK por algoritmo tradicional, devido as várias possibilidades apresentadas na mudança de estado, a listagem 8 mostra a implementação.

```
for (i=0; i<nControles; i++){
    ajustaControle(i);
    if (controle[i] == INDETERMINADO){
       return ERRO;
    }
}
if (controle[0] == 1){     // nível alto: guarda último valor de entrada
   meioj = entrada[0];
   meiok = entrada[1];
   controleAnterior = controle[0];
 }
if ((controleAnterior == 1) && (controle[0] == 0)){ // borda de descida
   if ((meioj == 0) && (meiok ==0)){     // memória
      saida[0] = saida[0];
   }
   if ((meioj == 0) && (meiok ==1)){
      saida[0] = 0;
   }
   if ((meioj == 1) && (meiok ==0)){
      saida[0] = 1;
   }
   if ((meioj == 1) && (meiok ==1)){
      saida[0] = !saida[0];
```

```
    }
    saida[1] = !saida[0];
    controleAnterior = controle[0];
}
return OK;
```

Listagem 8. O ff-JK.

Para a máquina de estados, criou-se um algoritmo exclusivo. Adotando o exemplo da figura 20 e tabela 2 do capítulo 4, que é uma máquina com uma entrada, quatro estados e três saídas, será explicado o algoritmo utilizado. Os índices da tabela são compostos pelo estado atual e pela entrada, e o conteúdo da tabela deve ser composta pelos próximos estados e pelas saídas.Para compor o índice usa-se o estado atual como os **bits** mais significativos e a entrada como os menos significativos (parte a da listagem 9). Para obter o próximo estado e o valor da saída a partir da valor decimal obtido da tabela, basta transformá-lo em binário e atribuir os valores convertidos seqüencialmente às saídas (listagem 9, parte a).

```
// parte a: cálculo do índice para a tabela
indice = 0;
for (i=0; i<nEntradas; i++){
    indice += entrada[i]  * (unsigned char) pow(2,i);
    estado *= 2;            //desloca um bit para a esquerda (estado
entrada)
}
indice += estado;
//parte b: acesso à tabela e conversão do valor em binário
int tmp = tabela[indice];    //valor decimal obtido da tabela
for (i=0; i<nSaidas; i++){
    saida[i] = (unsigned char) tmp % 2; //o resto da divisão e o valor do
bit
    tmp = (int) tmp / 2);    //o resultado da divisão é o prox.estado
}
estado = tmp;          //proximo estado
return OK;
}
```

Listagem 9. A máquina de estados.

6.2.3. CIRCUITOS PARA ARQUITETURA DE COMPUTADORES

Esta área é representada quase completamente por macro-funções que podem ser livremente definidas pelo usuário.

O usuário define num arquivo tipo texto, as tabelas de saída das macro-funções por ele projetada. Além da tabela projetada, deve constar neste arquivo um nome para cada tabela, e é justamente por este nome que a busca é feita dentro do arquivo.

```
nome
tamanho
valores para as saídas
```

Listagem 10. Formato do arquivo de tabelas.

Na instanciação da classe, o usuário ao parametrizar o objeto fornece o nome correspondente a macro constante do arquivo de tabelas. A operação, é definida por um ponteiro associado dinamicamente, pois, quando o algorítmo encontra o nome da macro procurada, ele carrega a tabela correspondente para a memória e a associa ao ponteiro (**ptrTabela*).

Apesar de visualizar-se que para estas macro-funções, a maior aplicação é na área de arquitetura de computadores, o usuário poderá construí-las visando qualquer área.

Como exemplo podemos construir a tabela para um somador binário, que poderia ser aplicado por exemplo na construção do computador SAP-1, exemplificado no capítulo 4. Mostramos a tabela para apenas um **bit**, porém, pode facilmente ser expandido para o número de entradas necessárias.

índice		conteúdo		
A	B	soma	vai um	valor decimal para a tabela
0	0	0	0	0
0	1	1	0	2
1	0	1	0	2
1	1	0	1	1

Tabela 8. Somador binário.

6.2.4. CIRCUITOS ANALÓGICOS

Como discutido na análise, matematicamente, bastam as quatro operações básicas e um método numérico para implementar o integrador, optou-se por implementar estas

operações no corpo da classe, desta forma, para cada uma das classes derivadas de ***analógicos***, existe um método que realiza a função associada a classe.

Voltando à estrutura apresentada na figura 30, nota-se que a classe ***analógica***, difere das demais apenas pelo tipo dos seus vetores *entrada* e *saída*, que contém valores *reais* (contínuos), e não discretos como nos digitais.

Para contornar esta diferença, utilizou-se para as entradas e saídas o tipo ***union*** pertinente a linguagem de programação C/C++, que permite manter-se na mesma localização de memória, dois tipos de dados diferentes, no caso, o tipo **byte** (**unsigned char**) dos componentes digitais e o tipo real (**double**) dos componentes analógicos.

Para esta área os sistema oferece já implementado os seguintes dispositivos: integrador; somador; subtrator; multiplicador e divisor cujos símbolos estão na figura 46 nesta ordem.

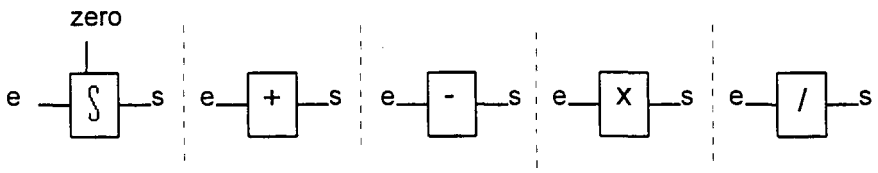


Figura 46. Símbolos dos dispositivos analógicos.

Nota-se que o integrador, possui uma entrada de controle chamada "zero", como este dispositivo tem uma função matemática acumuladora, em certos momentos pode ser interessante reinicializá-lo para prosseguimento da simulação, isto é feito atribuindo "um" para esta entrada. A listagem 11 mostra o código que implementa o método dos trapézios para cálculo de integral.

Como pode-se ver, análogo ao caso dos circuitos seqüenciais, este algorítmo é baseado no conceito de memória ou de realimentação. Para o cálculo são necessários o valor atual (*entradA[0]*) e o valor calculado anteriormente (*pontoAnterior*).

```
tempo++;
for (int i=0;i<nControles;i++)
    ajustaControle(i);
if (controle[0] == INDETERMINADO){
   saidA[0]=2;          //dispositivo desabilitado
   return OK;
}
if (controle[0] == 1){
   saidA[0] = 0;        //reinicializa integrador
}
else{                   //controle==0
   integral = pontoAnterior + entradA[0]; //método numérico
   integral *= dt;                        //para
   integral /= 2;                         //cálculo de integral
}
saidA[0] += integral;                     //resultado
pontoAnterior = entradA[0];               //ponto anterior para próx. cálculo
```

Listagem 11. Integrador.

6.2.5. CIRCUITOS PARA COMUNICAÇÃO COM O MUNDO REAL

Para implementar estes dispositivos, foram utilizadas funções da biblioteca do compilador C/C++. Estas funções são tais que acessam portas paralelas de comunicação do PC a partir de um endereço, que pode ser de uma controladora como da impressora, ou outra projetada pelo usuário, tanto para escrita como leitura.

O endereço é um parâmetro passado na instanciação do objeto.

Na leitura, os valores obtidos na porta de comunicação de entrada, são inicialmente colocados na cadeia *entrada* e após utilizados para atualizar a cadeia *saída* do dispositivo (listagem 12).

```
// leitura da porta
temp=_inp(endereco);          //lê a porta
for (i=0; i<8; i++)           //converte para cadeia de 8 posicoes
    entrada[i]=(temp&(unsigned char)pow(2,i));

//ajusta da cadeia saída
unsigned char portaEntrada::ajustaSaida(unsigned char indSaida){
for (i=0; i<8; i++){
    ajustaEntrada(i);
    saida[i] = entrada[i];
}
```

Listagem 12.A porta de entrada.

A função de escrita transfere para a porta de saída especificada os valores contidos em sua cadeia de *entrada* (listagem 13).

```
//escrita na porta
dado = 0;
for (i=0; i<8; i++).                    //conversão de uma cadeia de 8 posições
    dado += entrada[i] << i;            //para um byte de dados
dado = _outp(endereco, dado);           //o byte de dados é enviado para a porta
```

Listagem 13.A porta de saída..

Quatro portas foram implementadas: de entrada (1); de saída (2); de controle (3) e de protocolo (4), e os símblos utilizados pelo sistema são mostrados na figura 47.

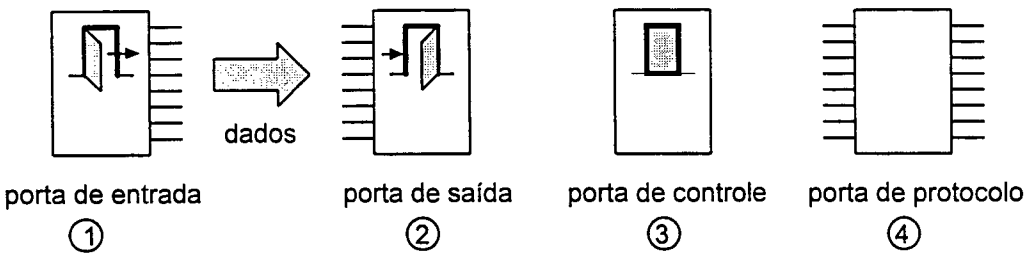


Figura 47. Desenhos das portas de comunicação.

A porta de controle é aplicada à interfaces que possuem o modo de programação atavés de um registrador controle. Desta forma o usuário, passa como parâmetros para o objeto, tanto o endereço da porta, quanto a palavra de controle, fato que torna desnecessárias as linhas para conexão de entrada e de saída (figura 47 parte 3).

A porta de protocolo pode ser utilizada em casos, onde a programação da interface a coloca em um modo, onde, uma das portas é utilizada para trocar sinais de protocolo (**handshaking**) entre o simulador e o ambiente físico. Estes sinais são necessários para manter a disciplina do fluxo do barramento. Além disso, muitas interfaces podem suportar a geração de interrupção, que seria feita através de uma destas linhas.

A implementação foi feita como se existisse duas portas no mesmo dispositivo, uma de entrada e outra de saída, (veja figura 48 e listagem 14) primeiramente é atualizado a

cadeia *entrada* e estes valores são então, enviados para a porta de comunicação. Num segundo passo a porta de comunicação é lida, e este dado serve para atualizar a cadeia *saida* do dispositivo. O circuito construído, fica então responsável pela interpretação e tratamento destas informações.

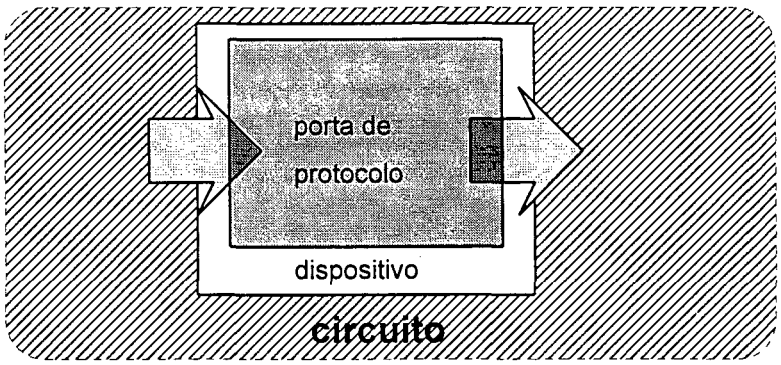


Figura 48. Representação em blocos da porta de protocolo.

```
//escrita na porta
dado = 0;
for (i=0; i<8; i++)                    //conversão de uma cadeia de 8 posições
    dado += entrada[i] << i;           //para um byte de dados
dado = _outp(endereco, dado);          //o byte de dados é enviado para a porta

// leitura da porta
temp=_inp(endereco);              //lê a porta
for (i=0; i<8; i++)               //converte para cadeia de 8 posicoes
    entrada[i]=(temp&(unsigned char)pow(2,i));

//ajusta da cadeia saída
unsigned char portaEntrada::ajustaSaida(unsigned char indSaida){
for (i=0; i<8; i++){
    ajustaEntrada(i);
    saida[i] = entrada[i];
}
```

Listagem 14. Porta de protocolo.

6.2.6. A CLASSE GERADORES

A classe ***geradores***, representa os objetos com *funções geradoras*. Como a sua classe mãe é abstrata, e nela não é implementada nenhuma função, também se torna abstrata, porém nela é definido o número de entradas como zero.

Os objetos aqui, são instanciados de suas subclasses, que incrementam atributos diversos, como *frequência*, no gerador de ondas quadradas, o valor da *saída anterior* (relativa ao valor corrente da saída) no gerador e na chave liga/desliga, e outros que auxiliam a implementação, que optamos por não enumerá-los dado código fonte no apêndice C.

Contrastando com outros dispositivos que necessitam atualizar suas entradas, este tem a cadeia de entrada nula e a cadeia de saída tem apenas uma posição.

Os geradores implementados foram divididos em duas áreas: digital e analógica.

Na digital: gerador de onda quadrada; gerador de corrente contínua e chave liga/desliga.

Na analógica: gerador de onda quadrada; de corrente contínua; das funções seno, exponencial, dente de serra.

O geradores de onda, necessitam de um algorítmo que implementa a devida oscilação em sua saída. Utilizou-se a seguinte técnica: A freqüência esta diretamente ligada à um parâmetro (um contador) que indica quantas vezes simulador passou por aquele dispositivo, ou seja, quantas vezes a lista foi percorrida. A cada "passada" este contador é incrementado e testado, caso ele tenha atingido o número desejado, a saída é alterada e o contador reinicializado. A freqüência mais alta possível, é atingida quando o contador atinge sua meta em todas as "passadas".

Os valores gerados pelos geradores de corrente contínua analógico podem ser acrescidos ou decrescidos de um valor parametrizado pelo usuário.

A diferença do gerador de onda quadrada e do gerador de corrente contínua, em relação aos da área digital, está nos valores gerados. Na instanciação deles, o usuário parametriza os objetos com valores que podem variar dentro da faixa de valores do tipo **double**.

O gerador de onda quadrada, solicita ao usuário a frequência, a amplitude máxima e a amplitude mínima, que pode ser positiva ou negativa, contrastando com os digitais que assumem somente os valores zero (0) e um (1).

De forma semelhante, o gerador de corrente contínua solicita o valor para ser gerado.

6.2.7. A CLASSE INDICADORES

Os dispositivos com *funções indicadoras* são pertinentes a classe ***indicadores***. Esta classe também possui derivadas, que assim como os geradores, apresentam sua função implementada na classe.

A tarefa consta então, de atualizar suas entradas e representá-las graficamente no monitor de vídeo, com o auxílio das funções gráficas fornecidas pela linguagem.

As saídas indicadoras também são implementadas dividindo-se em digital e analógica.

Na digital estão implementados os seguintes dispositivos: **led** e analisador lógico.

Na analógica o osciloscópio, que exemplificado por um trecho da sua implementação, na listagem 15, onde tempo é a coordenada do eixo x, e saidA e entradA as coordenadas do eixo y.

O analisador lógico e o osciloscópio, buscando similaridade com o objeto real, possuem pontas de provas, que são ligadas aos pontos onde se deseja analisar o resultado. Os resultados são visualizados em gráficos localizados no canto superior direito e inferior direito respectivamente.

```
_moveto_w(tempo-dtg,saidA[0]);//posiciona-se no último ponto marcado
_lineto_w(tempo,entradA[0]);//traça uma linha até o próx. pto.(interpolação)
```

Listagem 15. Funções que traçam o gráfico no osciloscópio.

6.3. A LIGAÇÃO DOS DISPOSITIVOS A ESTRUTURA GERAL DO CIRCUITO

A ligação entre dois dispositivos, é efetuada do objeto que recebe a informação (elemento receptor) para o objeto que fornece a informação (elemento gerador).

Esta medida foi adotada devido ao fato, de que ao simular um sistema, cada dispositivo será atualizado em suas entradas para que possa efetuar a transformação, e após a transformação atualizar a cadeia de saída.

Desta forma, sempre que um dispositivo for atualizar suas entradas, ele o fará de um dispositivo que já tem suas saídas atualizadas. Este método dá a conotação, de que um dispositivo vai buscar os valores para colocar em suas entradas, o que torna até natural ter um ponteiro de atributo no conjunto de entradas, que aponta para o elemento gerador.

Para melhor explicar os atributos citados nos itens anteriores, pode-se estudar o exemplo da figura 49. Este exemplo mostra um circuito composto por seis dispositivos: dois dispositivos da classes ***geradores*** (1 e 2), três da classe ***indicadores*** (4, 5 e 6) e um dispositivo hipotético (3) simbolizando a transformação ou operação do circuito.

O circuito fica, então, dividido em três partes: entrada; transformação e saída, o que confirma diretamente a teoria de sistemas.

A ligação entre os dispositivos é simbolizada, neste exemplo, por dois tipos de traços:

- pelos contínuos, representando os que a interface gráfica proporciona ao usuário;
- pelos pontilhados finalizadas por uma seta, representando a ligação estrutural efetuada pelos ponteiros.

As *entradas geradoras* (1 e 2) tem o atributo de entrada nulo, e as *saídas indicadoras* (4, 5 e 6) tem o atributo de saída nulo, devido sua funções.

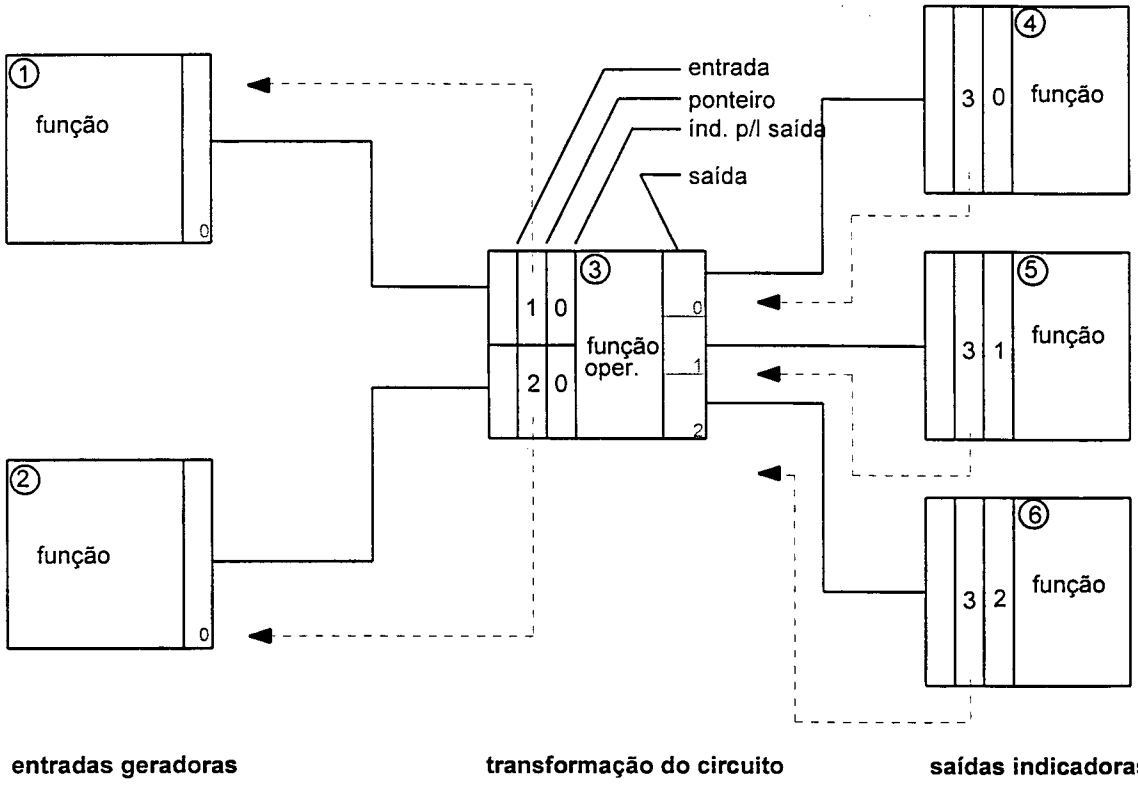


Figura 49. Exemplo de um circuito que mostra as ligações gráficas e estruturais

O dispositivo mais central (3) do desenho, possui tanto atributos de entrada, como de saída e representa a transformação do sistema.

Em relação à este dispositivo, a cadeia de duas posições chamada *entrada*, recebe os valores contidos na saída dos geradores (1 e 2) como indicado pelas linhas contínuas; na cadeia *ponteiro*, é retido em cada posição, o endereço do objeto que fornecerá os dados para suas entradas (na primeira posição o endereço do dispositivo número um e na segunda do dois).

Finalmente no atributo *índice para saída* encontra-se o *índice* correspondente à cadeia de valores da saída do objeto apontado. As duas posições da cadeia *índice par saída* do dispositivo número três, contém o valor zero, indicando a primeira posição da cadeia de saída dos dispositivos número um e dois respectivamente.

Sobre os valores contidos em sua *entrada* é efetuada a *função ou operação*, e após, ajustado os valores da cadeia de *saída* que neste exemplo é de três posições.

Assim sendo, os *dispositivos indicadores*, recebem em seu atributo *índice para saída* os índices 0, 1 e 2, segundo a seguinte explicação: o dispositivo quatro está ligado ao dispositivo três e receberá os dados da primeira posição do vetor de saída (índice 0); o dispositivo cinco, da segunda posição (índice 1) e o dispositivo seis, da terceira posição (índice 2).

6.5. CLASSES RELACIONADAS À ESTRUTURA DO PROGRAMA

Para a criação, construção e simulação de um circuito são utilizadas as classes ***circuito*** e ***gráfica*** que muitas vezes duplicam informações a respeito dos dispositivos, visto que, tratam respectivamente, da estrutura interna da simulação e da parte referente a apresentação gráfica.

Estas duas classes são apoiadas ainda, pelas classes: ***funções***; ***geral*** e ***menu***. Neste item serão discutidas a classe circuito e funções.

6.5.1. A CLASSE CIRCUITO

A principal estrutura desta classe é uma lista de ponteiros (figura 50), utilizada para armazenamento do endereço dos dispositivos criados durante o desenvolvimento do circuito. É sobre esta lista que são feitas todas referências e operações sobre os dispositivos.

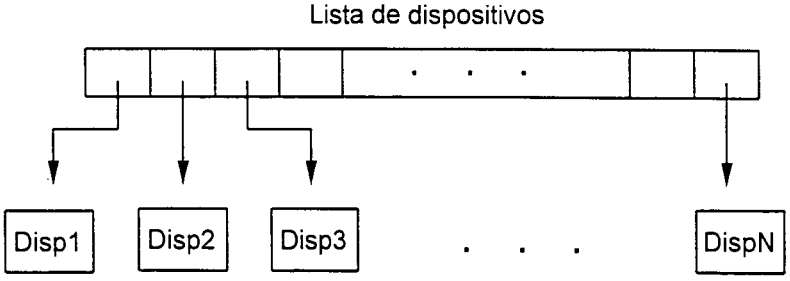


Figura 50. Lista de ponteiros para dispositivos

O tipo utilizado na lista é um tipo criado no programa, o da classe ***dispositivo***, porém, os objetos nela "guardados" são das diversas subclasses estudadas nos itens anteriores, como portas lógicas, decodificadores, multiplexadores, integradores, etc.

Isto só é possível, porque a lista, possui a característica abordada no paradigma da orientação a objetos e implementada na linguagem de programação utilizada (C++), o polimorfismo.

Esta lista é criada em na instanciação da classe circuito com um tamanho mínimo, e expandida dinamicamente, a cada inserção de novos elementos na construção do circuito.

A *ligação* e o *desligamento* dos componentes envolve, como foi dito, o trabalho conjunto de três classes: ***dispositivos***; ***circuito*** e ***gráfica***.

A ligação é feita pelo um método *meLigaCom* da classe ***Dispositivo***, atribuindo o endereço do objeto gerador ao campo *ptrPsaida*.

O desligamento acontece de forma similar, e o valor **NULL**[7] é atribuído no campo ponteiro para objeto gerador.

Durante a simulação o método *processaOtima*, percorre a lista otimizada enviando mensagens para a classe ***dispositivo*** ajustar a entrada e a saída dos dispositivos (veja listagem 16).

```
i = 0;
while ((listaOtima[i] != NULL) && (i<tamanhoOtima)){
    listaOtima[i]->ajustaEntrada(indEntrada);
    listaOtima[i]->ajustaSaida(indSaida);
i++;
};
```

Listagem 16. Parte do código do método que percorre a lista otimizada.

[7] **NULL** é uma constante do C/C++ que significa o valor nuio ou zero para um ponteiro.

Este é um processo que exige rapidez para poder fornecer o resultado dentro do menor tempo possível. Para se conseguir uma alta freqüência na simulação, a lista passa por modificações que visam uma *otimização* no circuito, no sentido de se acelerar o processo de simulação.

Esta otimização nada mais é que uma organização dos dispositivos dentro da lista, numa ordem que se dá no sentido do primeiro dispositivo a produzir um resultado diferente de indeterminado, ser o primeiro da lista, e da mesma forma para os próximos elementos, até atingir o final da lista original.

Para efetivar esta otimização, a lista passa por três fases antes da simulação em si:

1ª fase) verificar se todas as entradas, de todos os dispositivos, estão completamente ligadas à alguma saída de outros dispositivos;

2ª fase) organizar a lista de acordo com os tipos de dispositivos, primeiro vem os geradores, depois os dispositivos com funções modificadoras e por final os indicadores de resultados;

3ª fase) após esta organização, os dispositivos são processados um a um, e os que forem produzindo um resultado válido são colocados primeiro na lista, criando a seqüência ótima para fornecimento do resultado. A otimização final é obtido, geralmente, após várias "passadas" na lista.

Ao final, têm-se uma lista otimizada, com a combinação destas duas técnicas (segunda e terceira fases).

A segunda fase da organização, é primordial para a obtenção de um resultado correto na simulação de um circuito. Como exemplo toma-se um pequeno circuito, composto por um gerador de ondas quadradas e dois **led's**. Supor a ordem de instalação: ***led1***, *gerador*, ***led2*** como mostra a figura 51.a, e considerar uma simulação sem otimização.

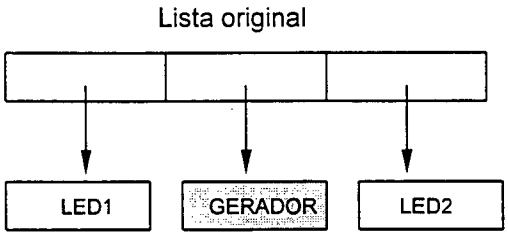


Figura 51.a. Exemplo de uma lista de dispositivos.

Quando esta lista for processada pela primeira vez (sem otimização), ocorrerá que o **led**1 será ajustado com o valor da saída do gerador, após isto, o gerador é processado mudando sua saída, o **led2**, sendo o próximo, receberá o valor complementado em relação ao **led1**. O usuário verá um **led** acesso e outro apagado, porém ligados ao mesmo gerador.

Após a organização da lista isto não mais ocorrerá, pois, a nova seqüência é **gerador, led1, led2** (figura 52).

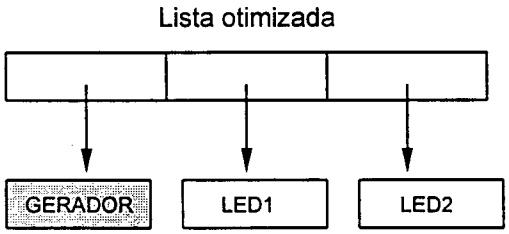


Figura 52. Lista de dispositivos após a instalação.

Num plano inicial, cogitou-se em utilizar técnicas e algoritmos para percorrimento de grafos, porém, buscando a simplicidade e praticidade, desenvolveu-se este algoritmo que forneceu um excelente resultado no percorrimento deste tipo de lista.

Sobre esta lista, é ainda realizada uma medição da velocidade de processamento computacional, para determinar a *freqüência de simulação do circuito*, já que isto vai variar de máquina para máquina. A técnica utilizada é bastante simples. Verifica-se o relógio de tempo real, processa-se o circuito otimizado em média 1000 (mil) vezes , e novamente verifica-se o relógio. Por estes valores, pode-se então obter em média, o tempo gasto para se processar o circuito uma vez.

Através deste tempo obtém-se a freqüência máxima de processamento do circuito. Está freqüência varia então com a velocidade de processamento da máquina utilizada, com o tamanho do circuito e finalmente com a complexidade da execução das funções dos dispositivos instalados.

Calculado este valor o usuário conhece a freqüência máxima de simulação do circuito, que também vale para os geradores, e pode estabelecer uma freqüência para a simulação. Fato que contribui para a análise do circuito, devido a facilidade de visualização dos resultados em baixa freqüência.

Esta rotina foi implementada e testada exaustivamente, sem obter-se, porém, os valores constantes esperados. Outros testes foram feitos em rotinas escritas em assembly, acontecendo os mesmos problemas. Ficando adiada para uma outra versão, a resolução deste problema.

6.6. CLASSES RELACIONADAS À INTERFACE COM O USUÁRIO

6.6.1. A CLASSE GRÁFICA

Esta classe permite, em sua função básica, a composição visual do circuito.

Conforme mostra na figura 53, cada objeto é representado no vídeo, por uma caixa com pontos visuais (pequenos traços) que servirão para conexão com outros objetos, esta interface foi implementada de forma semelhante ao FlowLearn [17].

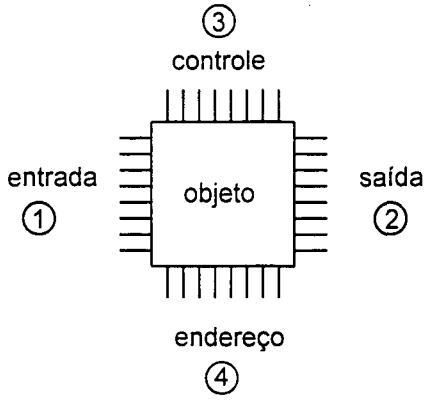


Figura 53. Um objeto com todos os traços possíveis para efetuar as ligações.

Existem quatro grupos para ligações:

- entradas de dados (1) apresentada do lado esquerdo;
- a saída (2) do lado direito;
- a entradas de controle (3) visíveis na parte superior da caixa;
- as entradas de endereço (4) na parte inferior.

Os objetos são instalados no circuito, de acordo uma matriz pré-definida, fixada no monitor de vídeo, conforme a figura 54. A utilização dessa matriz, introduz uma disciplina na construção de circuitos que facilita o trabalho do usuário.

Cada posição da matriz admite a instalação de apenas um objeto, que por sua vez possuem pontos de ligações são fixos.

Existe um cursor se movimenta de elemento em elemento desta matriz. Este cursor se apresenta na forma e tamanho de uma caixa, porém com linhas tracejadas.

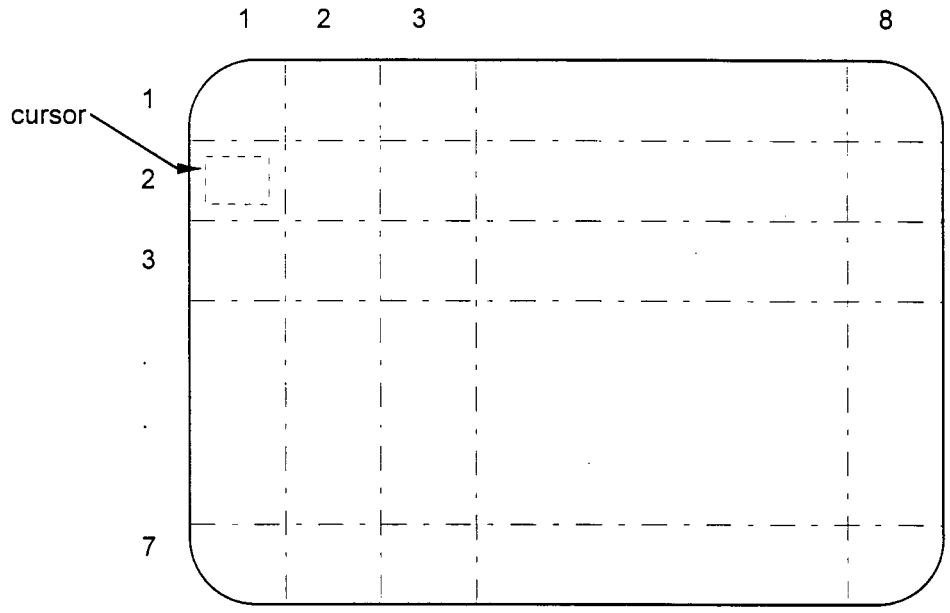


Figura 54. Matriz no monitor de vídeo.

As linhas desenhadas com traço-ponto, formando uma grade na matriz, não são visíveis ao usuário, estão apenas ilustrando o desenho.

A interligação dos dispositivos é representada por uma linha conforme mostrado na figura 55, e inicia no dispositivo a receber o dado (objeto 1), e vai até o dispositivo que fornece o dado (objeto 2).

Esta linha é desenhada pelo usuário através das setas do teclado. Para facilitar a condução desta linha e evitar erros na aproximação do dispositivo, ela "caminha" no vídeo segundo um passo igual a distância dentre os pontos de ligação. Opcionalmente pode-se manter a tecla Ctrl e manipular a setas, obtendo-se assim um passo triplicado.

Durante a ligação entre os dispositivos a classe ***gráfica*** recolhe os números dos dois dispositivos, das respectivas entradas e saídas que serão associados as posições das cadeias de entrada e saída de cada dispositivo.

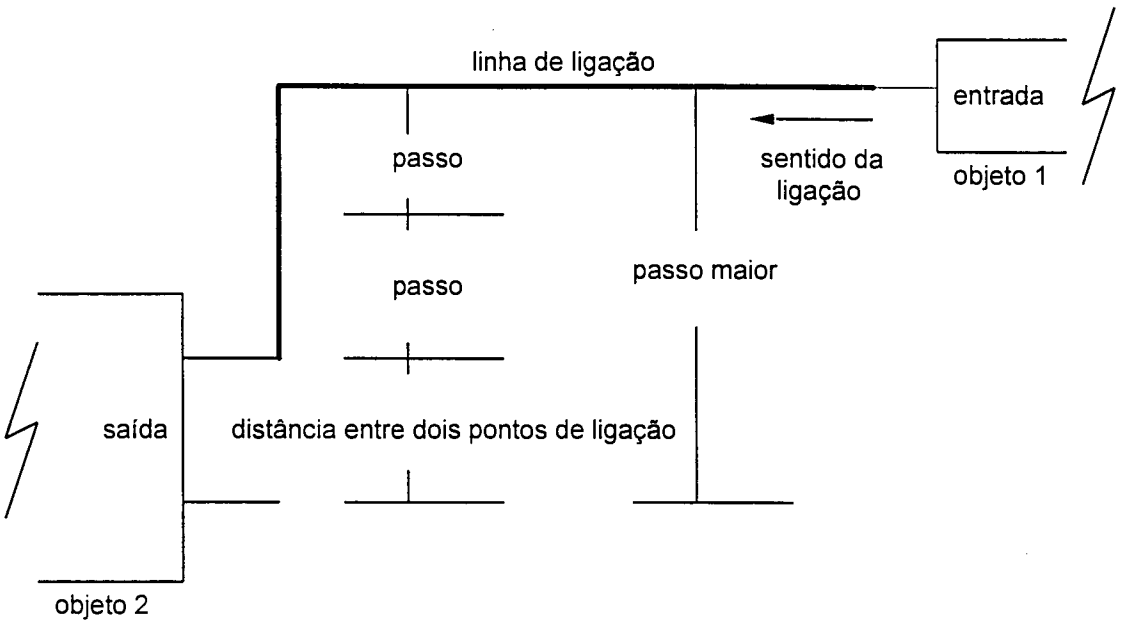


Figura 55. Relação entre os pontos ligação e o passo de incremento da linha de ligação.

Além da construção iterativa do circuito, a classe gráfica fornece serviços para *salvar* um circuito pronto (ou não) e *abrir* ou *carregar* um sistema armazenado em disco.

No desenvolvimento destes serviços, a classe gráfica grava em um arquivo as informações necessárias para descrever um circuito. Através destas é possível instanciar os objetos, reinterligá-los, associá-los com suas funções, enfim, reconstruir o circuito com seus dispositivos para uma nova simulação.

A forma e as informações armazenadas para cada dispositivo em seqüência no arquivo são mostradas na listagem 17.

```
nome do dispositivo
linha da matriz, coluna da matriz = número do dispositivo, código do dispositivo
cor do dispositivo
n° de entradas, n° de saídas, n° de entradas de controle, n° de entradas de endereço
parâmetro 1, parâmetro 2                                              (digitais)
parâmetro 3, parâmetro 4, parâmetro 5, parâmetro 6, parâmetro 7       (analógicos)
lista de coordenadas que formam as ligações 1
...
lista de coordenadas que formam as ligações n
```

Listagem 17. arquivo que armazena as informações do circuito

Deve-se constatar que o número de listas de coordenadas que formas as ligações é igual ao número de entradas do dispositivo, visto que, todas as ligações são feitas a partir da entrada para a saída.

6.6.2. A CLASSE GERAL

Esta classe constitui-se de um apanhado de métodos utilizados por todas as outras classes.

Pode-se destacar os métodos utilizados para entrada de dados, que foram desenvolvidos para suprir algumas deficiências nas funções fornecidas pela biblioteca da linguagem C++ utilizada.

Estes métodos desenvolvidos, permitem que usuário forneça, através de campos de edição mostrados em janelas parâmetros como *nome para o dispositivo*; *número de entradas*; *número de saídas*; etc.

Em resumo desenvolveu-se um *editor de linha* no ambiente gráfico utilizado, com características que facilitam a entrada de dados pelo usuário.

Outro método de destaque é o que permite a leitura de teclas com código estendido (p.ex. F1..Fn), e também a composição de códigos através do pressionar simultâneo da tecla **alt** e um número via teclado numérico, por exemplo a combinação da tecla **alt** com a seqüência 56, gera o código 56.

Com funções mais ilustrativas, cita-se métodos responsáveis por informar ao usuário a data e hora do sistema.

6.6.3. A CLASSE MENU

Esta classe implementa os menus janelados em modo gráfico, através dos quais o usuário e o sistema interagem.

Utilizou-se três tipos de menu: fixo numa barra horizontal (**popup**); flutuante no ambiente de trabalho (**dummy popup**) e caixa de diálogo (**dialog box**).

Os primeiro tipo, mostrado na figura 56, está representado por uma barra mais escura. e fica visualmente à disposição do usuário o tempo todo. Para acessá-los, basta pressionar a tecla correspondente (veja anexo C - Manual do Usuário). Para cada opção, em geral existe um novo menu.

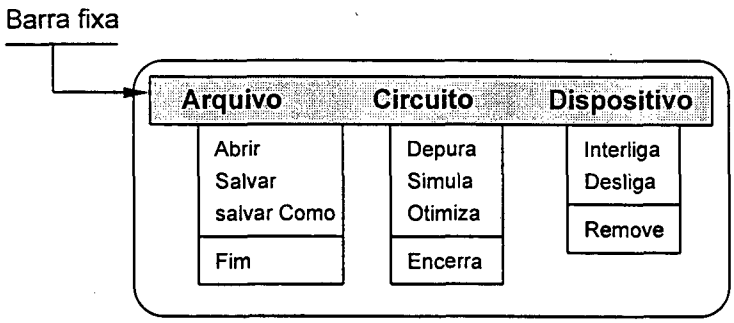


Figura 56. Exemplo de menu fixo na barra superior.

O segundo tipo, mostrado na figura 57, aparece somente quando o usuário o invoca, por exemplo, pela tecla **Enter**. Neste sistema ao se pressionar esta tecla visualiza-se o menu de áreas. Para cada opção existe pelo menos um novo menu, indicado pelas linhas pontilhadas.

Como pode-se ver, existe um menu principal (quadro maior), a partir de onde o usuário seleciona a área do componente a ser instalado, a partir daí, são oferecidos os menus específicos até se chegar ao dispositivo desejado.

Muitos dispositivos, requerem parâmetros para que sejam totalmente definidos.

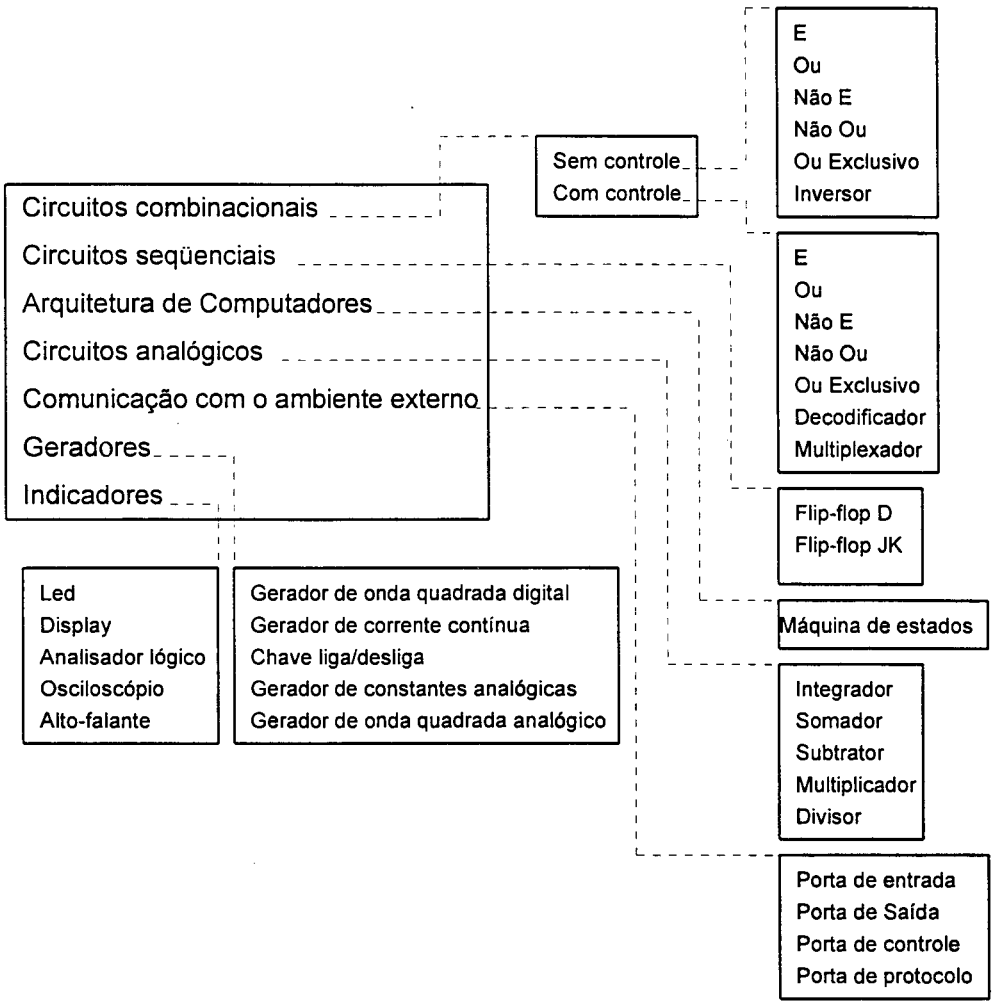


Figura 57. Hierarquia dos menus de iteração com o usuário

De forma semelhante, aos menus descritos acima, existem menus de entrada de dados (caixas de diálogo), como mostrados no exemplo da figura 58 parte a e b. Através de um editor de linha, constante na classe ***geral***, o usuário pode escrever o nome do dispositivo, o número de entradas e outras informações solicitadas.

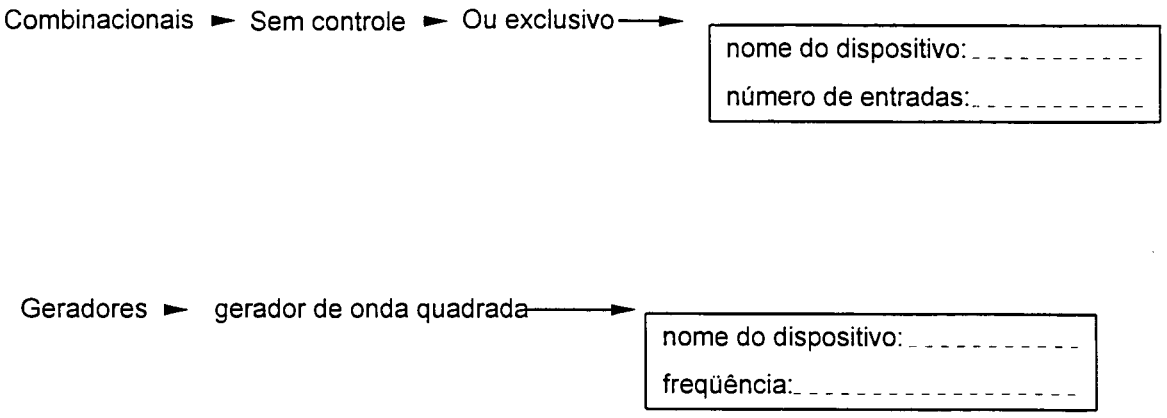


Figura 58. Exemplo de uma caixa de diálogo

6.7. SEQÜÊNCIA PARA CONSTRUÇÃO DE UM CIRCUITO E RELACIONAMENTO ENTRE AS CLASSES

Ao invocar o ***lab***GRAF, ele inicia o programa principal, que coloca o usuário em contacto com a tela do ambiente gráfico que é uma instância da classe ***gráfica***. A partir dai o programa principal controla todas as ações através dos comandos enviados via teclado (veja figuras 39 a 41).

Do ponto de vista do usuário a seqüência é a seguinte:

Através das setas o usuário escolhe uma posição no vídeo e invoca o ***menu*** escolhendo neste o dispositivo desejado, define seus parâmetros e o instala. A classe ***gráfica*** desenha o componente na tela. O componente é instanciado e ***circuito*** recebe uma mensagem do programa principal, com as características do dispositivos a ser instalado como parâmetros, e finalmente o insere na lista de dispositivos.

Instalados os dispositivos, passa-se a fazer as ligações.

Novamente através da classe ***gráfica***, que é por onde o usuário tem todo os acessos, faz-se o desenho da linha de ligação. Isto é feito a partir da entrada do dispositivo receptor para a saída do gerador. Ao final desta ação, a classe ***gráfica*** contém as seguintes informações: o número dos dois dispositivos que estão sendo ligados, os índices para a cadeia

de entrada e para cadeia de saída. Agora a classe ***gráfica*** pode enviar uma mensagem para ***circuito*** contendo todas estas informações, que são traduzidas para o endereço do objeto, a partir da lista de componentes. De posse da lista de ponteiros, ***circuito*** por associação, acessa o método específico de ***dispositivos*** completando a ligação estrutural.

Após todo o circuito ter sido montado, o usuário pode escolher, no menu da barra superior (ou a tecla rápida), a opção de otimização e de simulação. Isto significa o acionamento de métodos específicos da classe circuito para efetuar esta tarefa, até que o usuário envie uma nova mensagem ordenando a parada da simulação.

7. ANÁLISE DOS RESULTADOS

O objetivo deste capítulo, é de comparar, avaliar e criticar os resultados teóricos e experimentais atravé do ***lab***GRAF

A tabela 2 mostra um resumo comparativo entre este projeto e os sistemas estudados. Esta tabela baseia-se em critérios de avaliação adequados para o presente trabalho, eventualmente, algumas informações sobre sistemas existentes estão incompletas por falta de documentação referente.

	d/a	Interface gráfica	aberto ao usuário	dirigido ao ensino	orientado a objetos	permite criar novos objetos	interface de **hardware**	observações
labGRAF	**d/a**	**sim**	**sim**	**sim**	**sim**	*	**sim**	**dissertação**
LÓGICO	d	não	sim	sim	não	sim	não	Brasil
LOGICLAB	d	sim	sim	sim	sim	sim	não	incompleto
LOGSIM	d/a	sim	não	sim	?	não	não	Alemanha
ORCAD	d	sim	não	não	?	sim	não	EUA
ECE	d	sim	não	sim	?	não	não	EUA
CAT PACK	d	sim	não	sim	?	não	não	EUA
FLOWLEARN	a/d	sim	não	sim	sim	não	sim	Alemanha
LABTECH	d	sim	não	não	sim	não	não	EUA
SIMSCRIPT	d/a	não	não	não	não	sim	sim	EUA

Tabela 2. Comparação dos sistemas estudados.

A primeira coluna mostra os nomes dos produtos. A segunda mostra se o sistema fornece subsídios para sistemas digitais e analógicos. Por interface gráfica (terceira coluna), entende-se um ambiente onde o usuário é apoiado por ferramentas gráficas desde o início da montagem do circuito, até a visualização dos resultados.

A abertura ao usuário significa o código fonte a disposição do mesmo. Na quinta coluna da tabela, denota-se o tipo de produto, ou seja, se não é direcionado para o ensino, entende-se que deva ser para semi-profissional ou profissional.

Buscou-se visualizar nos **softwares** aplicativos se a implementação utiliza o paradigma orientado à objetos, naturalmente, os pacotes fechados escondem, além de outras,

esta informação. Os pontos de interrogação indicam onde não foi possível identificar a técnica de programação utilizada.

A próxima coluna, está relacionada com a possibilidade de adicionar, de alguma forma, novos objetos, ou mesmo novas classes às já existentes. Através do ambiente de simulação, pode-se parametrizar um objeto de forma que este seja diferente de outro instanciado pelas mesmas vias, por exemplo, uma porta lógica **E** com duas entradas diferenciando de outra porta lógica **E** com três entradas. São considerados objetos diferentes, porém, tem a mesma função. Esta coluna considera a possibilidade de criação de novos objetos com novas funções, e neste trabalho esta capacidade esta em fase de implementação.

Quanto a penúltima coluna, observa-se que além do ***lab***GRAF, apenas o FlowLearn e o SimScript oferecem, ainda que de forma diferente como discutido no capítulo 3, a possibilidade de comunicação do simulador com o ambiente físico. Foi desenvolvido o projeto (apêndice B) de uma interface de comunicação, a qual foi construída e testada, obtendo-se um resultado adequado no procedimento de aquisição/envio de dados. Esta interface, incentiva tanto o aprendizado de eletrônica, como de automação industrial que estuda sistemas automáticos. Este projeto pode ser utilizado como exemplo, onde, são fornecidas informações suficientes para o aluno desenvolver sua própria interface, projetando características adicionais, sofisticando este projeto ou criando o seu próprio.

Finalmente, a última coluna, observa o País origem e outras informações. Na primeira linha consta a palavra "dissertação", referenciando este trabalho. A terceira linha, referente ao produto LOGICLAB, consta a palavra incompleto. O estudo feito durante o capítulo 2, envolveu a implementação do código fonte fornecido no mesmo artigo publicado pelo autor. O resultado foi um bom ambiente gráfico, com menus fixos e flutuantes, sem porém, ter nenhum objeto para se testar.

Com base nesta comparação teórica/experimental fundamentada na pesquisa desenvolvida, pode-se julgar o ***lab***GRAF como um sistema de simulação adequado para aplicações do ensino.

A abordagem das funções implementadas através de tabelas, contribui tornando mais fácil a criação de novos objetos na área de simulação de sistemas digitais.

A integração das áreas digital, analógica e de comunicação com o ambiente externo, num só ambiente de trabalho, contribui de forma significativa para melhorar o ensino de disciplinas constantes no currículo dos cursos da área de computação, bem como de outros cursos, como os de engenharia e física.

O ***lab***GRAF é limitado em alguns pontos descritos a seguir:

- todo circuito pode ter no máximo cinqüenta e seis dispositivos;
- os circuitos devem ser montados através do suporte do teclado;
- para imprimir o circuito o usuário deve previamente ter carregado o programa graphics do DOS e pressionar a tecla PrtScr.

Estas limitações devem-se aos seguintes fatos:

- não foi desenvolvido nenhum algoritmo de rolamento de tela (**scroll**), desta forma, o tamanho do monitor de vídeo delimitou a matriz utilizada em sete linhas e oito colunas (7x8);
- não utilizou-se de algoritmos para ler dados do **mouse**, **joystick** ou outro dispositivos de entrada que não o dispositivo de entrada padrão do PC (o teclado), por considerar-se este um bom método de interfaceamento com o usuário, visto que, a interface gráfica oferece caminhos pré-definidos a serem seguidos pelo cursor, proporcionando um desenvolvimento do circuito de forma disciplinada e organizada;
- a comunicação deste sistema computacional com a impressora não foi desenvolvida.

7.1. Resultados obtidos no *lab*GRAF

Passa-se a enumerar alguns exemplos de circuitos físicos simulados no *lab*GRAF.

a) Um somador binário de quatro bits

Este somador efetua a operação entre duas palavras de quatro dígitos binários, conforme a figura 59, representados por oito chaves: a primeira palavra pelas chaves com números 10, 12, 14 e 16 e a segunda pelas chaves 18, 20, 22 e 24.

Os **led's** 1, 2, 4 e 6, representam os "vai um" (**carry**) de cada dígito e os 26, 28, 30 e 32 o resultado da soma entre cada digito das duas palavras.

O restante de dispositivos formam a lógica necessária para efetuar estas somas. A lógica utilizada é de quatro somadores inteiros.

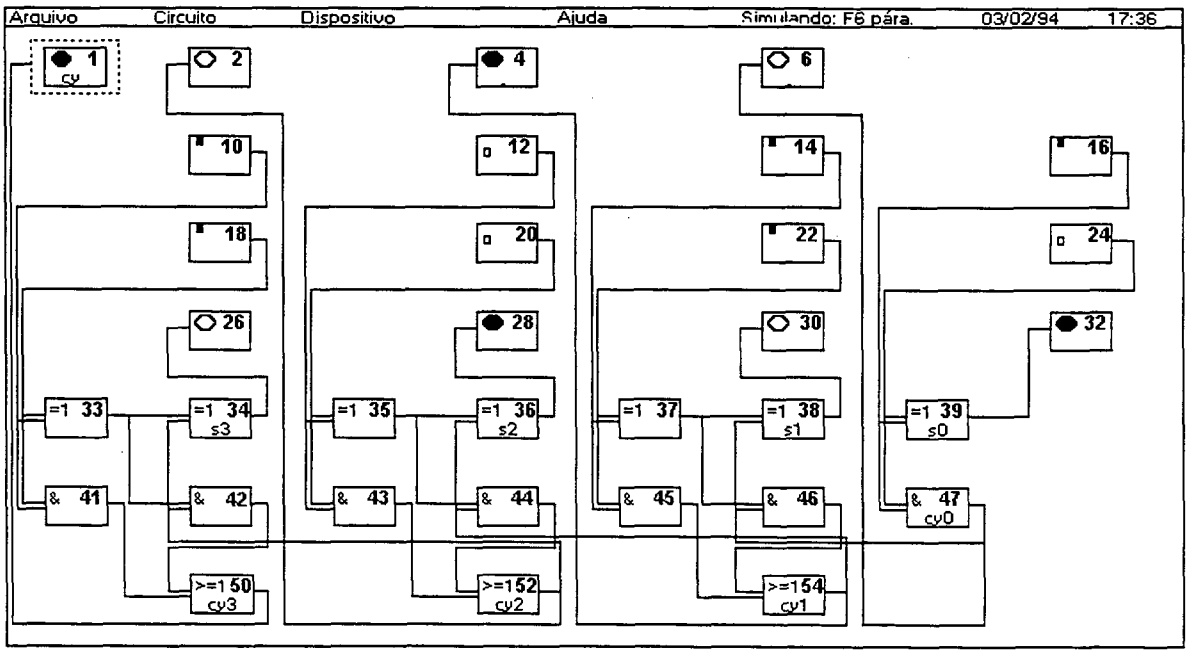


Figura 59. Tela com o exemplo do somador de 4 bits.

Após pressionar F6, para entrar no modo de simulação, pressione alt_*número da chave* para alterar os estados das mesmas. No exemplo da figura algumas chaves aparecem ligadas.

Para alterar o estado, por exemplo, da chave número 16, efetue a seguinte seqüência de teclas: mantenha a tecla alt pressionada e pressione a tecla 1 soltando-a em seguida, pressione a tecla 6 soltando-a em seguida e finalmente solte a tecla alt. Esta ação deve evidenciar a chave 16. Neste momento pressione a "seta para cima" para ligar a chave e a "seta para baixo" para desligar a chave.

b) Porta de comunicação do PC

Para comunicação do simulador com o ambiente externo ao computador, desenvolveu-se dispositivos que simulam portas lógicas.

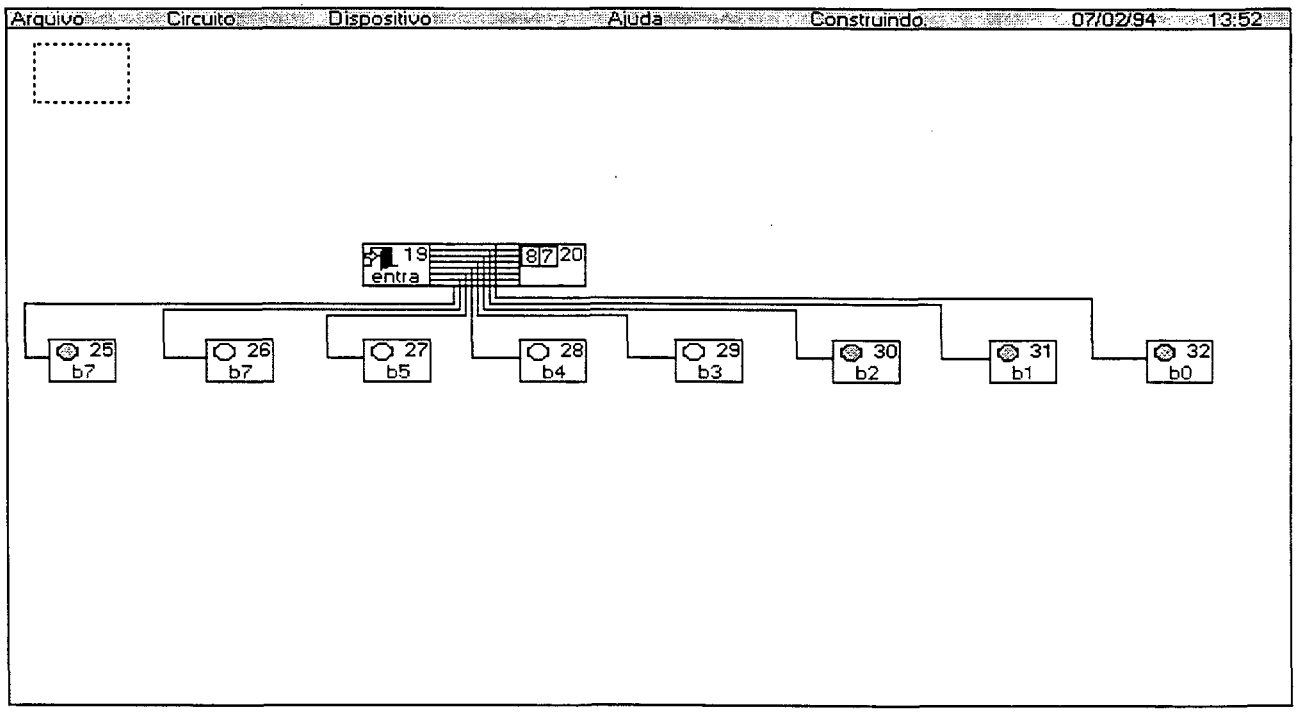


Figura 60. Porta de comunicação do PC

A porta paralela da impressora pode ser utilizada para entrada e saída de dados, o usuário poderá efetuar esta experiência sem a necessidade de um circuito externo específico para tal, utilizando uma impressora.

A figura 60, mostra uma porta lógica de entrada, que está conectada a porta de **status** da impressora. Esta porta é utilizada para informar o estado atual da impressora: ligada/desligada; com papel ou não, etc.

Para visualizar o resultado o usuário pode provocar estas mudanças de estado na impressora. Entre no modo de simulação (F6) e passe a ligar ou desligar a impressora, por e tirar o papel e assim por diante.

O resultado é mostrado nos **led's** 25 a 32 e também num **display** hexadecimal de 2 dígitos, cujo número é 20.

c) Um sistema de primeira ordem

O circuito RC discutido no capítulo 4, é agora utilizado como exemplo de implementação. Adotou-se o valor 1 (um) para a constante de tempo 1/RC, simulado pelos geradores número 26 e 34, os quais podem ser alterados pelo usuário para ilustrar a experiência.

A chave número 18 simula a função de entrada que excita o sistema, neste exemplo, um função pulso para V(t). Após entrar no modo de simula (F6), pressione a seguinte seqüência: alt_18, a seta "para cima", o que simulará o fornecimento de energia para o sistema e o gráfico no osciloscópio corresponderá à carga de um capacitor, que, dependendo do tempo que receba energia tenderá a saturação e o gráfico se estabilizará em um patamar, dependente dos parâmetros do sistema.

Desligando a chave 18 ("seta para baixo"), o sistema deixa de receber energia, e o osciloscópio mostrará o gráfico da descarga do capacitor.

A figura 61, mostra esta experiência efetuada de forma que a chave 18 tenha sido ligada e desligado dentro de um curto espaço de tempo, provocando o gráfico onde pode-se observar tanto a carga como a descarga do capacitor.

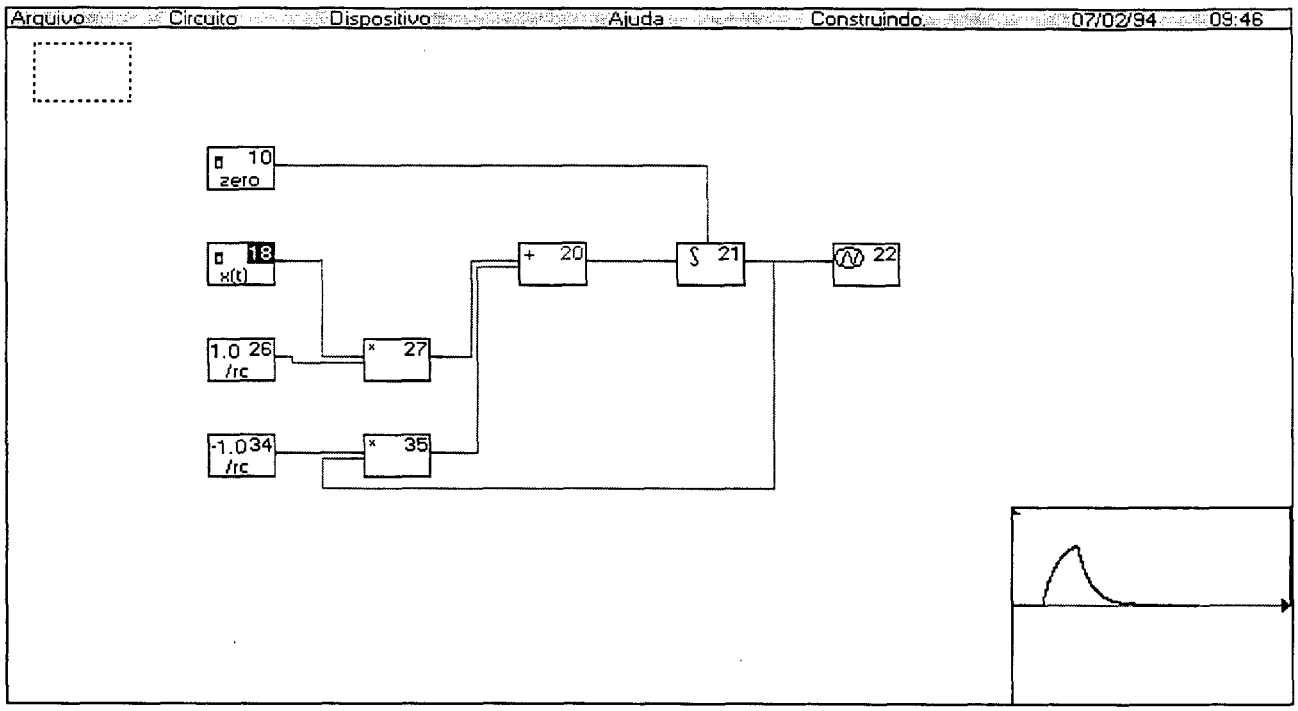


Figura 61. Tela com o exemplo do circuito RC.

Vale lembrar que muitos sistemas encontrados na natureza podem ser modelados através de uma equação diferencial de primeira ordem. Optou-se pelo exemplo de um circuito RC por se tratar de um exemplo clássico.

d) Um sistema de segunda ordem

Este exemplo também abordado no capítulo 4.

Neste exemplo (figura 62), coincidentemente, a função que alimenta o sistema é simulada pela chave número 18.

Usou-se este, para simular um sistema massa/mola/amortecedor (ou atrito) determinado pelas constantes simuladas pelos geradores números 26, 34 e 42, que podem ser alterados em tempo de simulação.

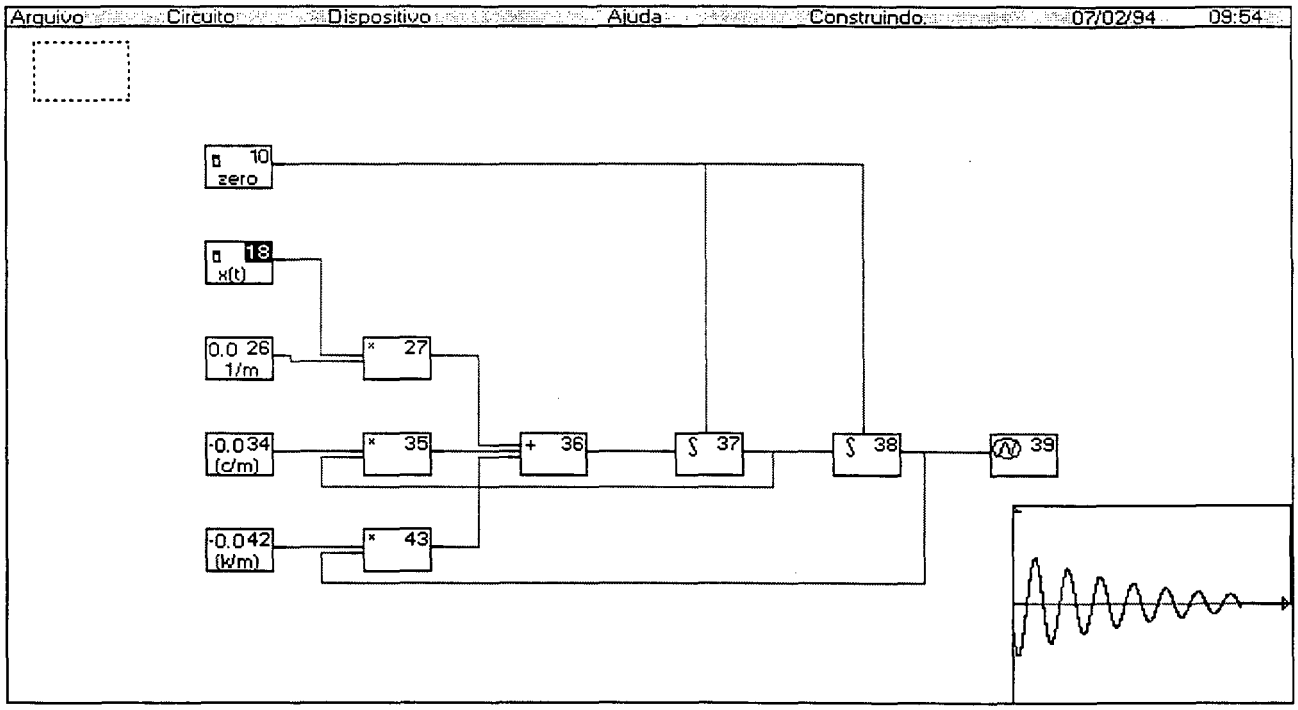


Figura 62. Tela com o exemplo do sistema massa/mola.

De forma bastante semelhante ao exemplo anterior o usuário pode obter o gráfico mostrado na figura 62, e também outros alterando o tempo de fornecimento de energia para o sistema e os parâmetros dos geradores citados no parágrafo acima.

O fornecimento de energia para este tipo de sistema, pode ser interpretado como um "puxão" que o experimentador aplica a mola para verificar a oscilação da massa pendurada na mesma, simulado aqui ligando momentaneamente a chave 18. O resultado depende da distância que a mola é esticada, bem como da constante de elasticidade e do atrito (ou grau de amortecimento).

8. CONCLUSÃO

No início deste trabalho apresentou-se um estudo, enfatizando que a automação de laboratórios didáticos, representa uma grande evolução no ensino superior.

Analisou-se o modelo existente para o ensino prático, e concluiu-se que a utilização de simulação no ensino, traz vantagens tanto para as universidades quanto para os alunos.

Estudou-se alguns sistemas existentes e concluiu-se que a grande maioria é direcionada à aplicações adversas ao ensino, não tem um ambiente de simulação homogêneo e em apenas dois casos encontrou-se a possibilidade de interfaceamento de **hardware**, porém, com alto custo.

Mostrou-se as semelhanças entre os sistemas discretos e analógicos, e sobre estas propôs-se um sistema homogêneo em relação às estas duas áreas.

Apresentou-se a análise, o projeto e a implementação do **software** e do **hardware** de um simulador de sistemas físicos. Preocupou-se em tratar o assunto em módulos didáticos viabilizando aplicações deste programa na escola de ensino superior. São eles: Circuitos Combinacionais, Circuito Seqüenciais, Comunicação com o Ambiente Externo, Arquitetura de Computadores e Circutos Analógicos.

Apresentou-se também, um estudo comparativo entre o sistema desenvolvido e alguns simuladores comerciais. Constatou-se que o ***lab***GRAF, ofereceu contribuições de relevante importância.

As principais características deste trabalho, se resumem como segue:

- em um tratamento unificado e homogêneo de sistemas discretos e contínuos sob um único ambiente;
- no interfaceamento do simulador com o ambiente externo ao computador;

- nas contribuições didáticas, no que se refere ao aluno ter acesso às informações referentes ao código do programa e ao projeto da placa de interface.
- os circuitos projetados são gravados em disco, desta forma, o aluno não perde o projeto elaborado, e pode continuar suas experiências em casa.

A aplicação deste projeto justifica-se por:

- existe uma redução de custos significativa, tanto na implantação de laboratórios computadorizados, como na elaboração de projetos em laboratórios que utilizam simuladores como ferramenta;
- a simulação contribui para a concentração do aluno na solução do problema em si, contrastando com bancadas convencionais, que em geral, afasta o aluno do problema central;
- a utilização de dispositivos digitais e analógicos integrados num ambiente homogêneo, incentiva a prática das duas áreas e facilita a elaboração de projetos;
- a utilização de **software** e **hardware** integrados num só ambiente, introduz o aluno ao contacto prático com os componentes eletrônicos de forma suave, além de possibilitar o estudo de sistemas automáticos;
- a posse do código fonte e dados da interface, fornece ao aluno diversas possibilidades de expansão, que é limitada apenas pela sua criatividade;
- a utilização da orientação a objetos, introduz facilidades que contribui para o ensino dos elementos que compõem o circuito, visto a forma parametrizada que eles são criados.

A análise feita baseada na teoria de sistemas, proporcionou grandes facilidades no projeto e implementação, dado o cunho da orientação a objetos de todo o trabalho. Esta técnica permitiu a criação de um ambiente único de simulação, traz vantagens didáticas, por introdir uma forma de abordagem homogênea dos sistemas facilitando o ensino destes.

Neste trabalho, considerou-se sempre, a importância de criar uma estrutura que possibilite a implementação de todos os tipos de objetos encontrados na natureza, relacionados às áreas abordadas, pois, como uma das principais características do pacote, é a abertura em relação ao código fonte e projeto de **hardware**, o próprio usuário, principalmente o aluno do

curso de computação, que é o principal alvo, tem condições suficientes para incrementar o pacote com novos blocos, tanto de **software** como de **hardware**.

Acredita-se na contribuição para o ensino de determinados tópicos, facilitando a compreensão das matérias da área no curso de computação.

Apesar de ter-se criado um sistema completamente utilizável, ele tem obviamente, aspectos que poderiam ser melhorados. Um destes é certamente a introdução de uma rotina habilitando o **mouse** como meio de entrada.

Outros pontos que podem ser reavaliados são:

- a estrututa das classes e suas relações;
- o comportamento em tempo.

Em relação a trabalhos futuros pode-se sugerir:

-
- uma estrutura de classes mais simples, para facilitar a criação de objetos através de especificação pelo usuário;
- uma otimização do simulador em relação ao tempo, junto com um aumento dos recursos em **hardware** e **software** para o atendimento de interfaces reais pode adequar o simulador como instrumento de automação, capaz de comandar ou controlar processos reais

Apêndice A

MANUAL DO USUÁRIO

A.1. CARACTERÍSTICAS GERAIS

O **lab**GRAF é um simulador de sistemas digitais e analógicos, que integra num só ambiente a composição destes dois tipos de sistemas de forma homogênea e paralela.

Homogênea por tratar de forma semelhante os dispositivos digitais e analógicos e paralela por permitir a composição dos dois tipos de circuitos ao mesmo tempo no mesmo ambiente e até mesmo construir sistemas mistos.

Além disso, é possível comunicar-se com o ambiente externo ao computador, através de dispositivos virtuais de comunicação, utilizando alguma interface física.

O usuário é apoiado por uma interface gráfica, ilustrada na figura A.1, através da qual pode-se compor os circuitos e visualizar os resultados. Os circuitos podem ser construídos com apoio de menus (itens A.2 e A.3), ou através de teclas rápidas (item A.4).

Abaixo do menu está o ambiente de trabalho, e nesse o usuário pode digitar a tecla ENTER e visualizar um menu de instalação de dispositivos, que flutua de acordo com a posição do cursor.

O **lab**GRAF opera em computadores da linha IBM-PC ou compatíveis, com a configuração mínima possuindo o sistema operacional DOS, 640K de memória, um acionador de discos de baixa densidade (360Kb) e monitor EGA ou superior.

A partir do prompt do DOS, o comando **lab**GRAF invoca o sistema.

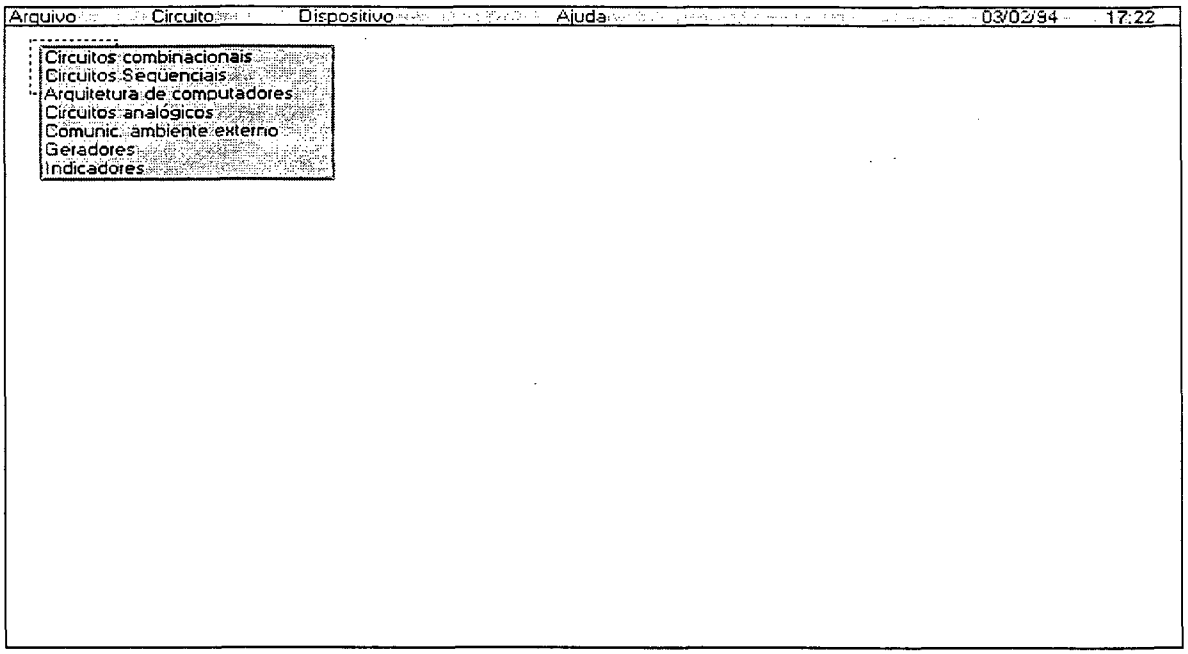


Figura A.1. Menu da barra superior e ambiente de trabalho com o menu flutuante sob o cursor.

A.2. O MENU DA BARRA SUPERIOR

a) Arquivo

Figura A.2. Menu arquivo.

Para acessar este menu use alt_a. Aqui pode-se: *abrir* (F3) ou carregar um arquivo com um circuito previamente construído e gravado em disco e *salvar* (F2) circuitos construídos com o mesmo nome ou com outro.

A última opção permite *abandonar* (alt_x ou alt_F4) o **lab**GRAF.

b) Circuito

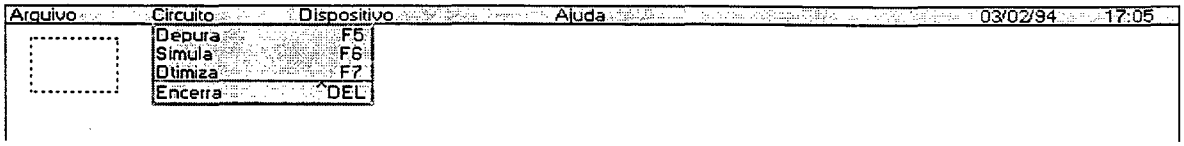


Figura A.3. Menu Circuito.

Através das teclas alt_c, este menu oferece todas as opções relacionadas a simulação de um circuito.

Depurar (F5) um circuito significa simular este passo a passo, ou seja, a cada toque da tecla F5 a simulação avança em um passo.

A próxima opção é para *simular* (F6) o circuito por tempo indeterminado, para parar usa-se novamente F6.

Otimizar (F7) significa criar uma lista de dispositivos com uma seqüência, que permita simular o circuito na mais alta freqüência possível. Isto é feito automaticamente na carga de um novo circuito, porém, é necessário que o usuário pressione F7, sempre que houver a construção de um novo circuito, ou a modificação de um existente.

A última opção deste menu é a de encerramento de um circuito. A opção *encerra*, limpa o ambiente de trabalho, para o início da construção de um novo circuito. A carga de um circuito armazenado em disco provoca esta ação automaticamente. Sempre que houver uma mudança deste tipo o sistema pede uma confirmação (item A.5).

c) Dispositivo

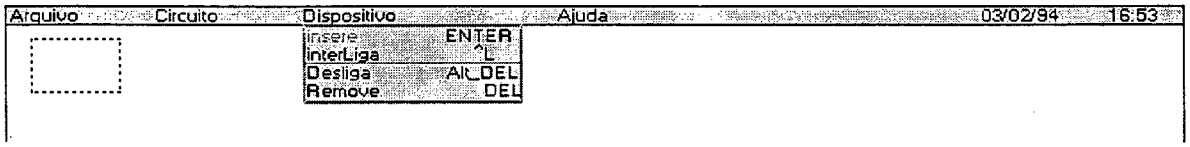


Figura A.4. Menu dispositivo.

Aqui constam as funções relacionadas com os dispositivos que comporão, ou que compõem o circuito. Use alt_d para acessar este menu.

Para efetuar a ligação entre dois dispositivos, posicione o cursor sobre o dispositivo que vai receber o dado do outro e pressione a opção de *interliga* (Ctrl_L), esta ação provoca a mudança para o modo de interligação de dispositivos. Em seguida pressione Ctrl_L seguidamente, para posicionar sobre uma das entradas deste dispositivo.

Neste modo é mostrado um cursor em forma de cruz, que se move ao comando das quatro setas. Mova o cursor até uma das saídas do dispositivo que vai fornecer o dado, esta ação desenhará uma linha na cor verde, simulando um fio elétrico que liga dois dispositivos. A ligação é finalizada automaticamente, quando o cursor chegar em uma saída do dispositivo alvo.

A operação inversa, ou seja, desfazer uma ligação, é feita posicionando-se apenas no dispositivo que recebe a informação e escolhendo-se a opção *desliga* (alt_DEL). A partir daí o usuário pode escolher qual entrada ele quer desligar pressionando seguidamente alt_DEL. Utilize a tecla DEL para consumar a remoção.

d) Ajuda

Pressione alt_j ou F1, para visualizar o menu de ajuda.

O item A.4. apresenta a tabelas com as informações do menu de ajuda.

A.3. O MENU FLUTUANTE

Para acessar o menu flutuante, use a tecla ENTER dentro do ambiente de trabalho. Esta ação mostra uma lista de subáreas de alguns cursos da área tecnológica.

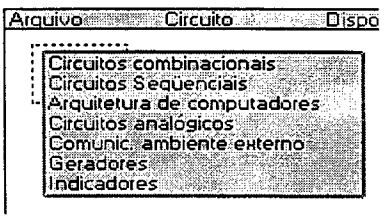


Figura A.5. Detalhe com o menu flutuante

Sob estas subáreas encontra-se todos os componentes constantes desse sistema computacional.

Existem duas maneiras de selecionar uma das áreas que aparecem neste menu:

- com auxílio das setas "para baixo" e "para cima" evidenciar pela cor "azul claro", uma das opções e ativá-la pressionando a tecla ENTER;

- digitar a tecla referente a letra evidenciada pela cor verde água.

Após o usuário escolher a subárea, sempre aparece um sub-menu, é nele que se escolhe o dispositivo desejado. Apenas na área de circuitos combinacionais, aparece um sub-menu intermediário com as opções de dispositivos "sem controle" e "com controle".

A instalação de um componente é feita na posição do cursor, que deve ser previamente localizado onde se deseja fixar o componente.

Passa-se a citar os dispositivos relacionados com cada área do menu.

a) Circuitos combinacionais

Sem controle: portas lógicas (E, OU, NÃO E, NÃO OU, OU EXCLUSIVO e INVERSOR).

Com controle: portas lógicas (E, OU, NÃO E, NÃO OU e OU EXCLUSIVO), decodificador 3x8 e multiplexador 4x1.

b) Circuitos seqüenciais

Latch's D e Flip-flops tipo D e JK.

d) Arquitetura de computadores

Máquina de estados.

e) Circuitos analógicos

Integrador, somador, subtrator, multiplicador e divisor.

f) Comunicação com o ambiente externo

Porta de entrada, porta de saída, porta controle e porta de protocolo.

g) Geradores

Este item é dividido em duas partes: digital e analógica.

Na digital tem-se:

- gerador de onda quadrada, que oscila entre os níveis lógicos "baixo" e "alto", segundo uma freqüência escolhida pelo usuário. Utilize alt_*número do gerador* para alterar a freqüência com as setas.
- gerador de corrente contínua, gera "baixo" ou "alto", sem oscilar;
- chave liga/desliga, semelhante à anterior porém o usuário pode alterar o valor em tempo de simulação. Utilize alt_*número da chave* para ligar ou desligar com as setas;
- chave comutadora, possui duas saídas, quando uma está "baixa", a outra está "alta". Utilize alt_*número da chave* para alterar os estados das saídas com as setas.

Na analógica:

- gerador de onda quadrada, oscila entre dois níveis determinados pelo usuário, que podem ser inclusive negativos. Utilize alt_*número do gerador* para alterar a freqüência com as setas.
- gerador de corrente contínua, gera um valor determinado pelo usuário. Utilize alt_*número do gerador* para alterar o valor da saída com as setas, entre dois valores determinados pelo usuário.

f) Indicadores

Da mesma forma que o anterior, existem dos digitais e analógicos.

Digitais:

- led, indica os níveis "alto" e "baixo" em sua entrada alterando sua cor de vermelho para branco, respectivamente;
- display, com oito entradas, converte os valores binários recebidos em um número hexadecimal, e o apresenta;

- analisador lógico, representa a seqüência de valores recebidos num gráfico que varia com o tempo.
- alto-falante; reproduz um "beep" se receber um nível "alto" em sua entrada.

Analógicos:

- osciloscópio; representa os valores recebidos em sua entrada num gráfico de duas dimensões;

A.4. AS TECLAS RÁPIDAS

Este item mostra uma tabela resumo com todas as teclas rápidas do sistema.

TECLA	FUNÇÃO
F1	Invoca o menu de **ajuda**, opcionalmente use alt_j
F2	**Salva** o circuito construído em um arquivo
F3	**Abre** ou carrega um arquivo gravado em disco
F4	**Salva** o circuito com outro nome que não o corrente
F5	**Depura** o circuito
F6	**Simula** o circuito por tempo indeterminado
F7	**Otimiza** a lista de dispositivos para simulação
F8	
F9	
F10	**Ativa** o menu superior
F11	
F12	
alt_0 a alt_56	Entra no **modo de alteração** do valor da saída dos geradores com os números indicados, em tempo de simulação
Ctrl_L	Entra no **modo de interligação** de dispositivos
4 setas (alteração da saída)	**Incrementa/decrementa** os valores da saída dos geradores
4 setas (menu superior)	**Alterna** entre os sub-menus do menu superior
4 setas (ambiente de trabalho)	**Movimenta** o cursor
4 setas (modo de interligação)	**controla** a ligação entre dois componentes num passo normal
Ctrl_4 setas	**Controla** o cursor no modo de interligação com um passo triplicado
alt_f	**Altera** o valor da freqüência do gerador sob o cursor
alt_s	**Altera** o valor da saída do dispositivo sob o cursor
alt_i	**Altera** o passo de integração do integrador sob o cursor
DEL	**Remove** o dispositivo sob o cursor
alt_DEL, DEL	**Posiciona** sobre e **Remove** uma ligação do dispositivo sob o cursor
Ctrl_DEL	**Apaga** o circuito corrente

A.5. AS MENSAGENS

Existem três tipos de mensagens: o que indica o estado do simulador em relação ao circuito, os de confirmação de tarefas e o que indica algum erro ocorrido na tarefa corrente.

O primeiro tipo aparece na barra do menu superior, e as mensagens são:

- construindo;
- otimizando;
- depurando;
- simulando.

O segundo aparece no ambiente de trabalho sob a forma de janela. São menus, que pedem a confirmação pelo usuário, para execução de alguma tarefa solicitada. As mensagens são:

- Remove dispositivo? (S/N)
- Abandona circuito atual? (S/N)
- Circuito modificado, salva? (S/N)
- Abandona o labGRAF? (S/N)

O terceiro aparece na parte inferior do vídeo para indicar os seguintes erros:

- Erro ao abrir arquivo *nome*.LAB;
- Arquivo *nome*.LAB não encontrado;
- Posição já ocupada.
- Dispositivo sem entrada para conectar;
- Não existe dispositivo para desfazer ligação;
- Não existe dispositivo para remover;
- Nem todas as entradas estão conectadas;
- Todas as entradas já foram conectadas;
- Não existem componentes ou circuito para otimizar;
- Não existe circuito ou ele não foi otimizado;

A.6. EXEMPLO DE CONSTRUÇÃO DE UM CIRCUITO

Este item trata o exemplo da construção de um circuito muito simples, visando a familiarização do usuário com ambiente do **lab**GRAF.

Em primeiro lugar use as quatro setas do teclado para localizar o cursor, na posição desejada para instalar o dispositivo. Por exemplo, a partir da posição original do cursor, canto superior direito, pressione duas vezes a tecla "para baixo" e duas "para a direita", em seguida pressione ENTER par obter o menu de áreas.

A partir deste menu escolha a opção **Geradores**, isto o levará um sub-menu contendo os possíveis geradores deste sistema. Escolha um gerador de onda quadrada digital, este pedirá um **nome** opcional, digitando ENTER ele será nulo, uma **freqüência máxima**, que será um fundo de escala para este dispositivo e finalmente a **freqüência** de atuação. Após parametrizado este dispositivo, aparecerá na tela um quadrado na cor lilás, com o número 19 no seu canto superior direito, um "v.u." no canto superior esquerdo e um nome, se fornecido pelo usuário, na parte inferior.

Movimente agora, o cursor uma posição para a esquerda e pressione ENTER, escolha a opção **Indicadores** e dentro do sub-menu a opção **led**. Será solicitado apenas o nome do dispositivo. O usuário visualizará outro quadrado na cor vermelha, contendo o número 20 no canto superior direito. No canto superior esquerdo, um círculo branco ou vermelho, indicando os níveis lógicos baixo alto, respectivamente, recebidos na entrada deste **led**.

Ainda com o cursor sobre o **led**, segure a tecla Ctrl apertada e simultaneamente pressione a tecla L (minúsculo) (Ctrl-L). Esta ação leva o usuário para o modo de interligação de dispositivos. A partir deste momento as quatro setas controlam um cursor em forma de cruz, que deve ser orientado até a saída do **gerador**, quando isto acontecer, automaticamente o **lab**GRAF sai do modo de ligação, porque este dispostivo tem apenas uma entrada.

Este circuito apesar de extremamente simples, deve passar por uma otimização. Isto é feita pelo simples pressionar da tecla F7.

Agora pronto para funcionar, o circuito pode ser simulado. Pressione F6 e visualize o **led** piscando de acordo com a freqüência do gerador.

Faça então, a experiência de alterar o valor da freqüência: segure a tecla **alt** apertada e pressione a tecla 1 (um) e em seguida a 9 (nove) soltando a tecla **alt** após isto (alt_19). Esta ação evidenciará o dispositivo número 19, que será visualizado com um quadrado vermelho sob seu número. Neste ponto utilize as setas "para baixo" e "para cima" diminuindo ou aumentando a frequência do gerador e observe o **led** piscando mais lento ou mais rápido respondendo a estas mudanças. Opcionalmente pode-se utilizar a teclas "+" ou "-" para obter o mesmo efeito.

A.7. OUTROS EXEMPLOS MAIS COMPLEXOS

a) Um somador binário de quatro bits

Este somador efetua a operação entre duas palavras de quatro dígitos binários, conforme a figura A.6, representados por oito chaves: a primeira palavra pelas chaves com números 10, 12, 14 e 16 e a segunda pelas chaves 18, 20, 22 e 24.

Os **led's** 1, 2, 4 e 6, representam os "vai um" (**carry**) de cada dígito e os 26, 28, 30 e 32 o resultado da soma entre cada digito das duas palavras.

O restante de dispositivos formam a lógica necessária para efetuar estas somas. A lógica utilizada é de quatro somadores inteiros.

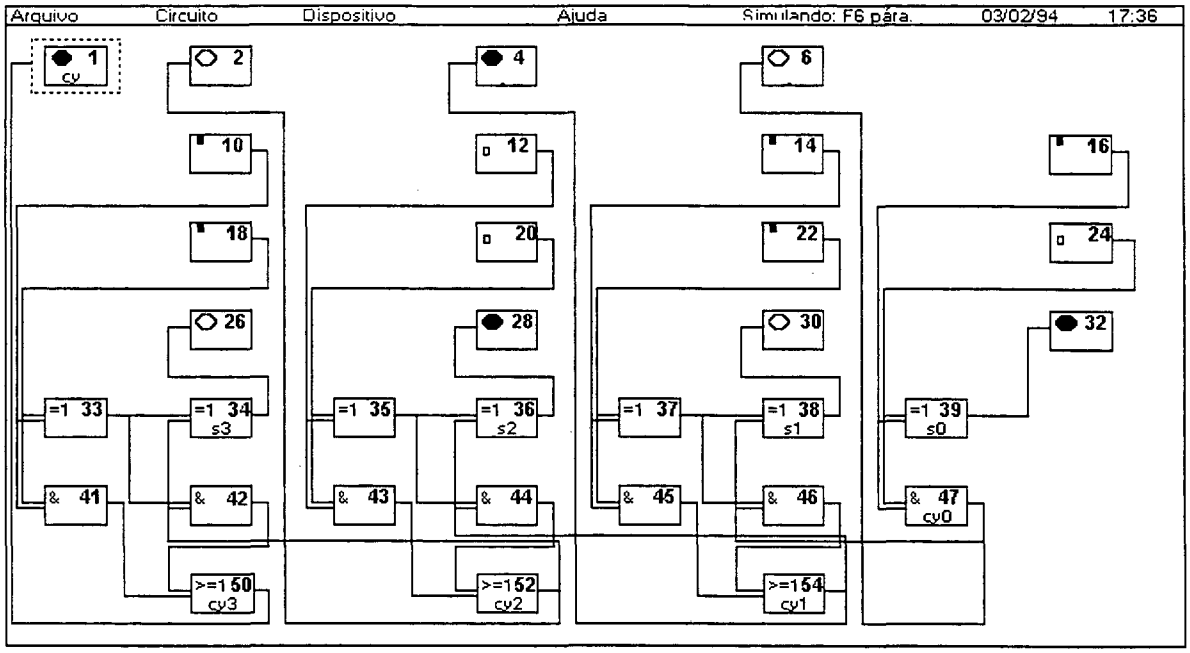


Figura A.6. Tela com o exemplo do somador de 4 bits.

Após pressionar F6, para entrar no modo de simulação, pressione alt_*número da chave* para alterar os estados das mesmas. No exemplo da figura algumas chaves aparecem ligadas.

Para alterar o estado, por exemplo, da chave número 16, efetue a seguinte seqüência de teclas: mantenha a tecla alt pressionada e pressione a tecla 1 soltando-a em seguida, pressione a tecla 6 soltando-a em seguida e finalmente solte a tecla alt. Esta ação deve evidenciar a chave 16. Neste momento pressione a "seta para cima" para ligar a chave e a "seta para baixo" para desligar a chave.

b) Porta de comunicação do PC

Para comunicação do simulador com o ambiente externo ao computador, desenvolveu-se dispositivos que simulam portas lógicas.

A porta paralela da impressora pode ser utilizada para entrada e saída de dados, o usuário poderá efetuar esta experiência sem a necessidade de um circuito externo específico para tal, utilizando uma impressora.

A figura A.7, mostra uma porta lógica de entrada, que está conectada a porta de **status** da impressora. Esta porta é utilizada para informar o estado atual da impressora: ligada/desligada; com papel ou não, etc.

Para visualizar o resultado o usuário pode provocar estas mudanças de estado na impressora. Entre no modo de simulação (F6) e passe a ligar ou desligar a impressora, por e tirar o papel e assim por diante.

O resultado é mostrado nos **led's** 25 a 32 e também num **display** hexadecimal de 2 dígitos, cujo número é 20.

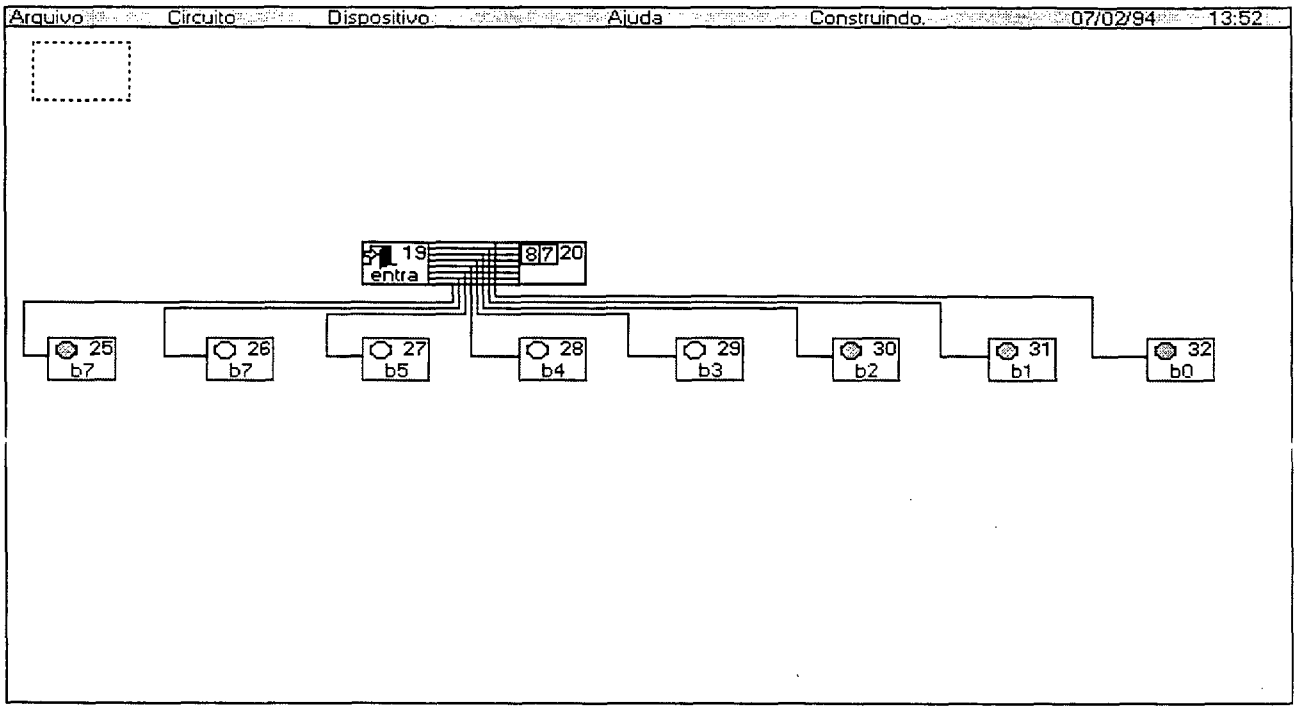


Figura A.7. Porta de comunicação do PC

c) Um sistema de primeira ordem

O circuito RC discutido no capítulo 4, é agora utilizado como exemplo de implementação. Adotou-se o valor 1 (um) para a constante de tempo 1/RC, simulado pelos geradores número 26 e 34, os quais podem ser alterados pelo usuário para ilustrar a experiência.

A chave número 18 simula a função de entrada que excita o sistema, neste exemplo, V(t). Após entrar no modo de simula (F6), pressione a seguinte seqüência: alt_18, a seta "para cima", o que simulará o fornecimento de energia para o sistema e o gráfico no osciloscópio corresponderá à carga de um capacitor, que, dependendo do tempo que receba energia tenderá a saturação e o gráfico se estabilizará em um patamar, dependente dos parâmetros do sistema.

Desligando a chave 18 ("seta para baixo"), o sistema deixa de receber energia, e o osciloscópio mostrará o gráfico da descarga do capacitor.

A figura A.7, mostra esta experiência efetuada de forma que a chave 18 tenha sido ligada e desligado dentro de um curto espaço de tempo, provocando o gráfico onde pode-se observar tanto a carga como a descarga do capacitor.

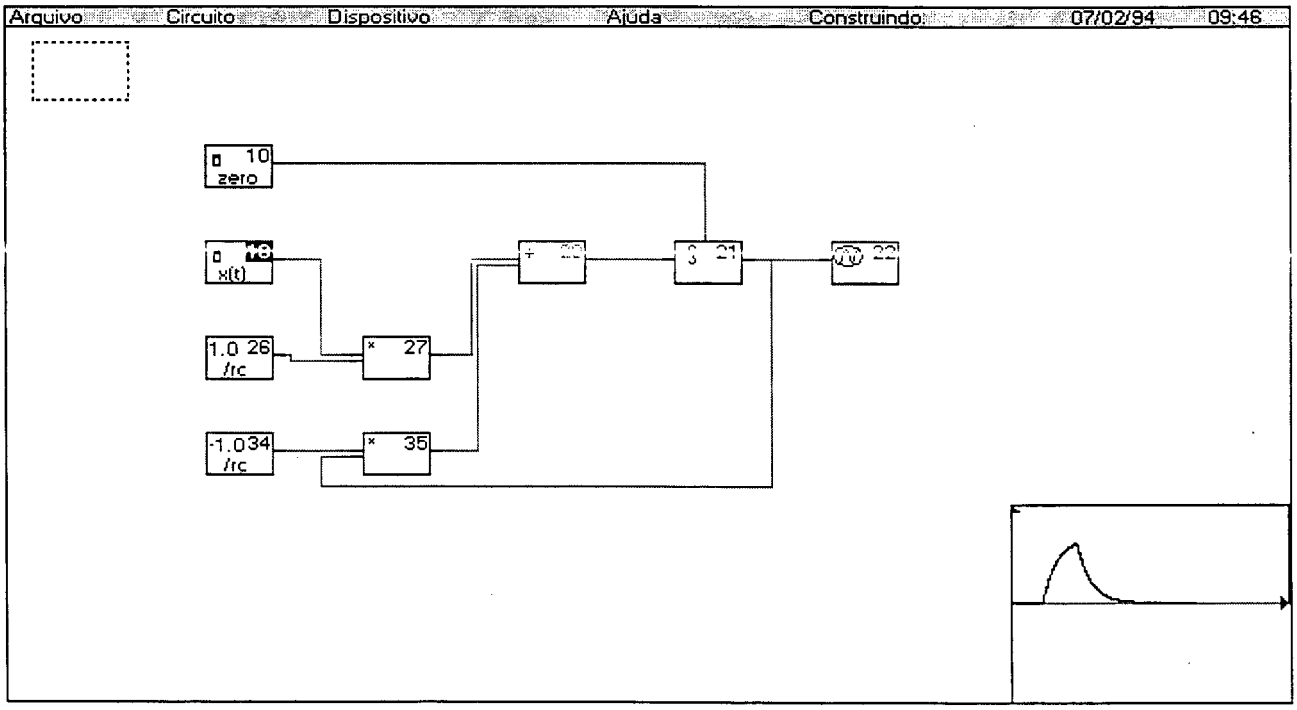


Figura A.7. Tela com o exemplo do circuito RA.

Vale lembrar que muitos sistemas encontrados na natureza podem ser modelados através de uma equação diferencial de primeira ordem. Optou-se pelo exemplo de um circuito RC por se tratar de um exemplo clássico.

d) Um sistema de segunda ordem

Este exemplo também abordado no capítulo 4.

Neste exemplo, coincidentemente, a função que alimenta o sistema é simulada pela chave número 18.

Usou-se este, para simular um sistema massa/mola com amortecedor (ou atrito) determinado pelas constantes simuladas pelos geradores números 26, 34 e 42, que podem ser alterados em tempo de simulação.

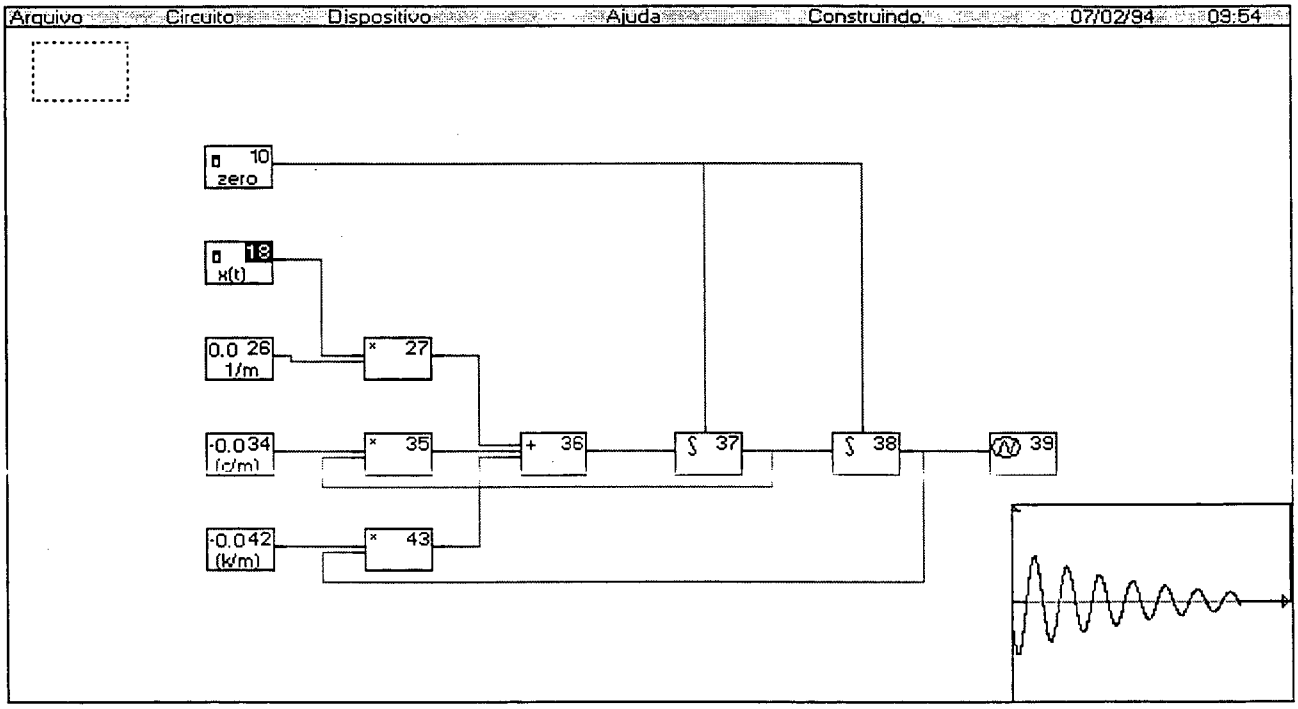


Figura A.8. Tela com o exemplo do sistema massa/mola.

De forma bastante semelhante ao exemplo anterior o usuário pode obter o gráfico mostrado na figura A.8, e também outros alterando o tempo de fornecimento de energia para o sistema e os parâmetros dos geradores citados no parágrafo acima.

O fornecimento de energia para este tipo de sistema, pode ser um puxão que o experimentador dá na mola para verificar a oscilação da massa pendurada na mesma, simulado

aqui ligando momentaneamente a chave 18. O resultado depende da distância que o mola é esticada, bem como da constante de elasticidade e do atrito (ou grau de amortecimento).

A.8. CONSIDERAÇÕES FINAIS

O **lab**GRAF foi desenvolvido para apresentar características abrangentes, buscando ser genérico de modo a atender áreas tecnológicas e científicas como computação, automação, física e engenharia.

O resultado foi um ambiente de simulação que atende a todos os tipos de pos sistemas físicos e com aplicação a diversos cursos universitários ou mesmo de segundo grau.

A versão atual é apresentada em um disquete de 360K, onde consta o **lab**GRAF (arquivo labgraf.exe mais dois arquivos *.fon) e alguns exemplos pré-elaborados pelo autor (arquivos *.lab).

Apêndice B

PROJETO DE UMA PLACA DE INTERFACE USANDO A PPI 8255

Para atender a área *comunicação com o ambiente externo* discutido durante o decorrer deste trabalho, faz necessário uma interface para traduzir as informações oriundas de dispositivos externos para o computador e vice&versa.

Um dos pontos altos da *abertura* deste sistema, é o fato de não se fazer qualquer restrição ao tipo de interface, desta forma, o próprio usuário pode projetar a sua interface.

Cabe à este apêndice, oferecer o projeto completo de uma interface, que poderá ser construída pelo usuário com facilidade, devido a sua extrema simplicidade. Esta facilidade não prejudica sua funcionalidade, visto que tem o 8255 como principal componente, e como pôde-se ver no apêndice A, ele oferece várias possibilidades de programação por **software**.

B.1. DECODIFICAÇÃO DE ENDEREÇOS

Normalmente o 8255 não requer lógica externa para interfacear dispositivos externos, porém, para acessá-lo existem apenas três linhas de entrada: A0, A1 e -CS, o que induz a construção de uma lógica adicional para decodificação de endereços, para atingir uma gama mais larga de possíveis endereços, e evitar sobreposições com endereços utilizados pelo PC.

A maneira mais simples de decodificar um endereço ou um grupo de endereços de E/S para um projeto de uma interface, é procurar um bloco de endereços não utilizados dentro espaço total, e contruir o sua própria circuitaria.

. O PC reserva os endereços em hexadecimal de 0000 à 01FF, para os dispositivos de suporte e para E/S do sistema da placa mãe. Além disso, os endereços 00C0 à

01FF não são utilizados como portas de entrada e saída desta placa, e podem portanto, ser decodificados para projetos de interfaces através do sistema de barramento.

O usuário pode projetar qualquer circuito lógico para decodificar um endereço ou grupo dentro deste espaço.

Existem três tipos de decodificador:

- o que tem um endereço fixo (figura B.1, parte 1);
- o de endereço fixo, porém, pode-se selecionar um dentre alguns possíveis através de chave (figura B.1, parte 2);
- e o mais sofisticado, tem vários endereços possíveis, prograds através de chaves e comparadores (figura B.1, parte 3)

Descarta-se a primeira opção, por ser de boa prática, adotar um circuito de decodificação que não se limite há apenas um endereço, aumentado as possibilidades de instalação da placa de interface. Entre as duas últimas opções, optou-se para este projeto pela segunda, que apesar de menos sofisticada, é a solução mais econômica e mais simples que a terceira.

O 8255 tem duas entradas (A0 e A1) para acessar quatro endereços. Além disso a interface deve ser habilitada ou selecionada pela linha -CS (**chip select**). A partir destas informações pode-se escolher o grupo de endereços que acessam a interface e construir o devido decodificador. Os endereços são seqüenciais a partir de uma *base*.

Por exemplo, se a porta A estiver no endereço 780H, a porta B, C e registrador de controle estarão seqúencialmente em 781H, 782H e 783H.

Adotou-se para esta interface o seguinte grupo de endereços *base*: 780; 784; 788; 78C; 790; 794; 798; 79C. Para cada uma dessas bases existem 3 outros formando uma seqüência.

Cada um desses endereços *base* são escolhidos exclusivamente, isto pode ser feito por um conjunto de **jumpers's**, um conjunto de chaves tipo **dip-switch** ou ainda, por uma chave rotativa. Neste exemplo, o decodificador é seguido de uma série de oito **jumper's**, para selecionar um endereço.

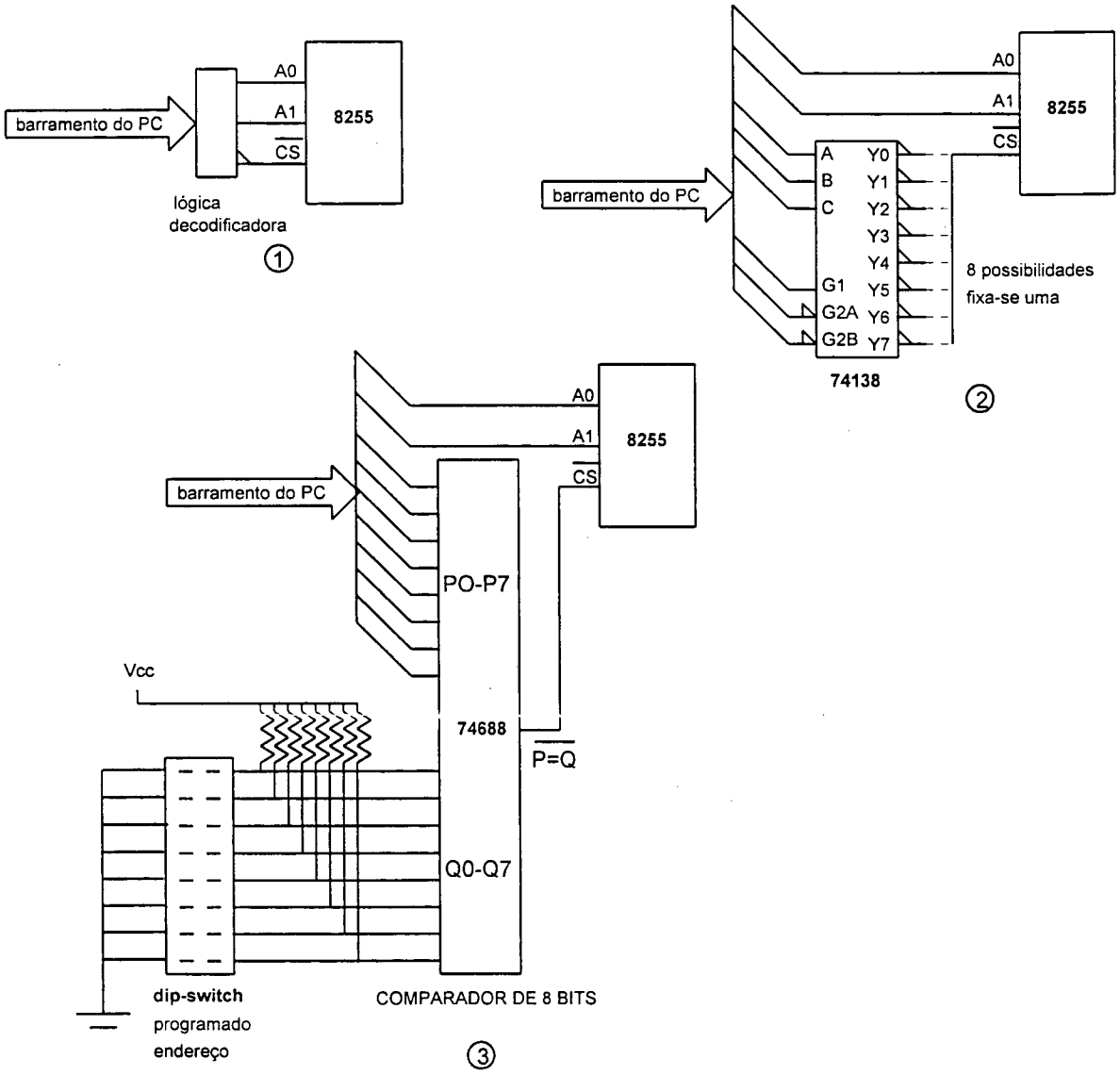


Figura B.1.Três opções para o decodificador de endereços.

B.2. O CIRCUITO

Composto por quatro componentes, incluindo o 8255, e um conjunto de **jumper's**.

O 74LS04 é um TTL composto por seis inversores.

O 74LS30, é formado por apenas uma porta lógica **NÃO E** de oito entradas.

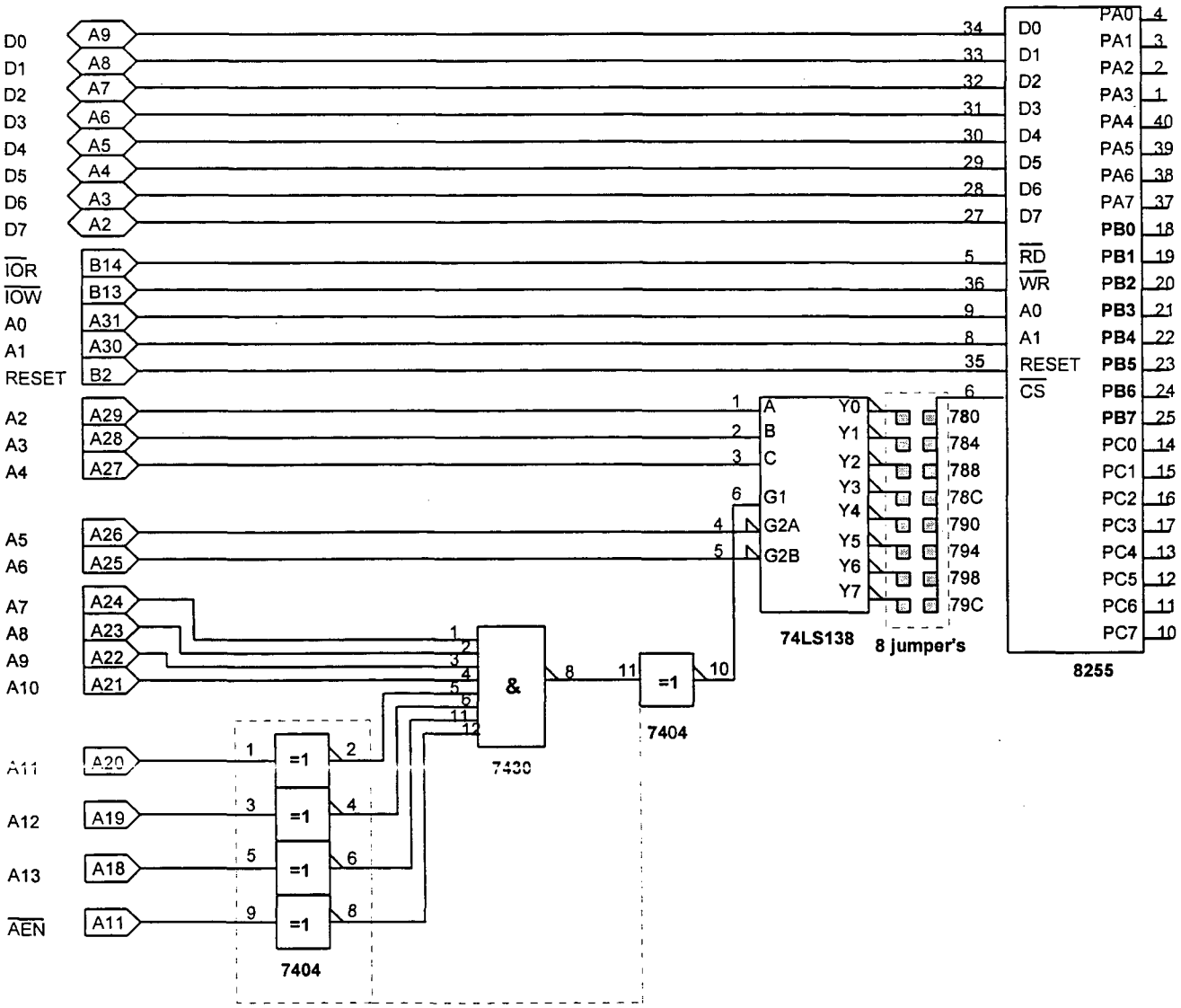


Figura B.2. Diagrama esquemático da placa de interface

O 74LS138, é um multiplexador com três entradas e oito saídas (3x8), possui po ainda três entradas de habilitação denotadas G1, -G2A e -G2B, para habilitar este dispositivo, estas estradas devem conter respectivamente 1, 0 e 0.

A tabela B.1 ilustra o barramento do PC e as possíveis palavras, para formar os endereços que acessam o 8255. Cada um deles é escolhido exclusivamente atavés de um **jumper**, e correspondem seqüencialmente às saídas Y0 a Y7 do multiplexador 74138.

A primeira linha corresponde ao número das linhas do **slot** do PC (veja figura B.3), a segunda o nome das linhas do barramento de endereços, correspondentes às do **slot**.

Deve-se ressaltar, que a inicial A, apesar de coincidente, são devido a fatores diferentes: o **slot** tem os lados A e B como mostra a figura B.3. e os nomes das linhas do barramento vem da palavra endereço, do inglês **Address**.

A11	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31	
AEN	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	780
0	0	0	0	1	1	1	1	0	0	0	0	1	0	0	784
0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	788
0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	78C
0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	790
0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	794
0	0	0	0	1	1	1	1	0	0	1	1	0	0	0	798
0	0	0	0	1	1	1	1	0	0	1	1	1	0	0	79C
0			7				8 ou 9				0 a C				

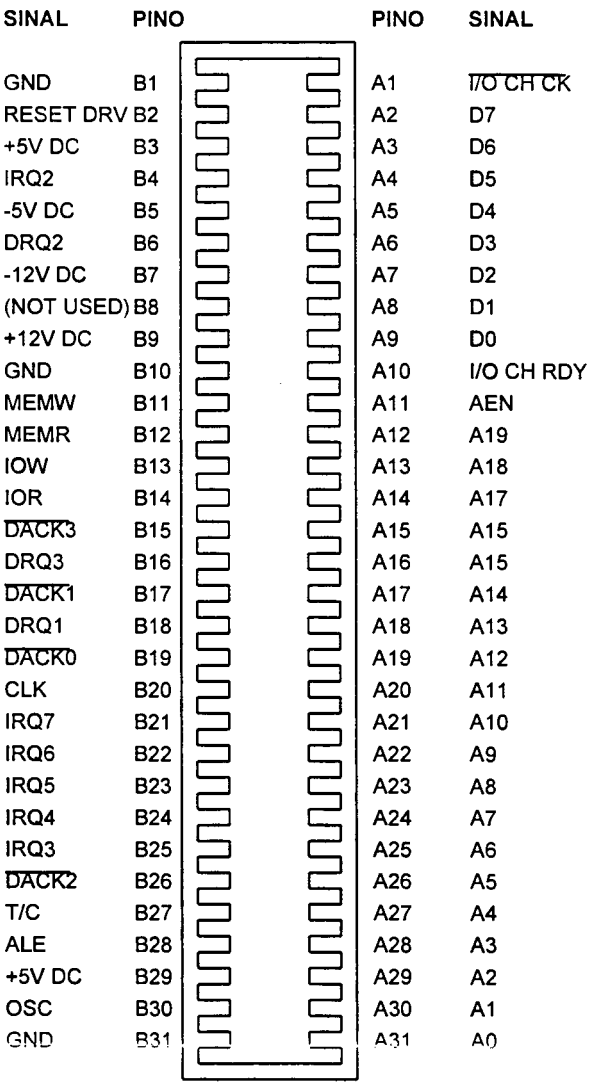
Tabela B.1. Composição dos endereços que acessam o 8255.

Cada endereço mostrado na tabela corresponde ao endereço da *porta A* do 8255 (endereço *base*), para acessar as portas *B*, *C* e *registador de controle*, basta escrever endereços subseqüentes.

B.3. O SLOT DO PC

A figura B.3 apresenta a configuração do **slot** do PCxt, ou seja 8 bits. Neste **slot** são encaixadas as placas de interface ou controladoras.

Estas placas devem ser confeccionada, preferencialmente, em placa de fibra de vidro dupla face. Suas dimensões devem ser tais, que a placa se encaixe perfeitamente num dos **slots** disponíveis no PC.



SINAL	PINO	PINO	SINAL
GND	B1	A1	$\overline{\text{I/O CH CK}}$
RESET DRV	B2	A2	D7
+5V DC	B3	A3	D6
IRQ2	B4	A4	D5
-5V DC	B5	A5	D4
DRQ2	B6	A6	D3
-12V DC	B7	A7	D2
(NOT USED)	B8	A8	D1
+12V DC	B9	A9	D0
GND	B10	A10	I/O CH RDY
MEMW	B11	A11	AEN
MEMR	B12	A12	A19
IOW	B13	A13	A18
IOR	B14	A14	A17
$\overline{\text{DACK3}}$	B15	A15	A15
DRQ3	B16	A16	A15
$\overline{\text{DACK1}}$	B17	A17	A14
DRQ1	B18	A18	A13
$\overline{\text{DACK0}}$	B19	A19	A12
CLK	B20	A20	A11
IRQ7	B21	A21	A10
IRQ6	B22	A22	A9
IRQ5	B23	A23	A8
IRQ4	B24	A24	A7
IRQ3	B25	A25	A6
$\overline{\text{DACK2}}$	B26	A26	A5
T/C	B27	A27	A4
ALE	B28	A28	A3
+5V DC	B29	A29	A2
OSC	B30	A30	A1
GND	B31	A31	A0

Figura B.3. Pinagem do **slot** e linhas de sinais do barramento

B.4. A CONSTRUÇÃO DA PLACA

As dimensões necessárias para construção da placa são mostradas na figura B.4, e para este projeto são até excedentes dado o pequeno número de componentes.

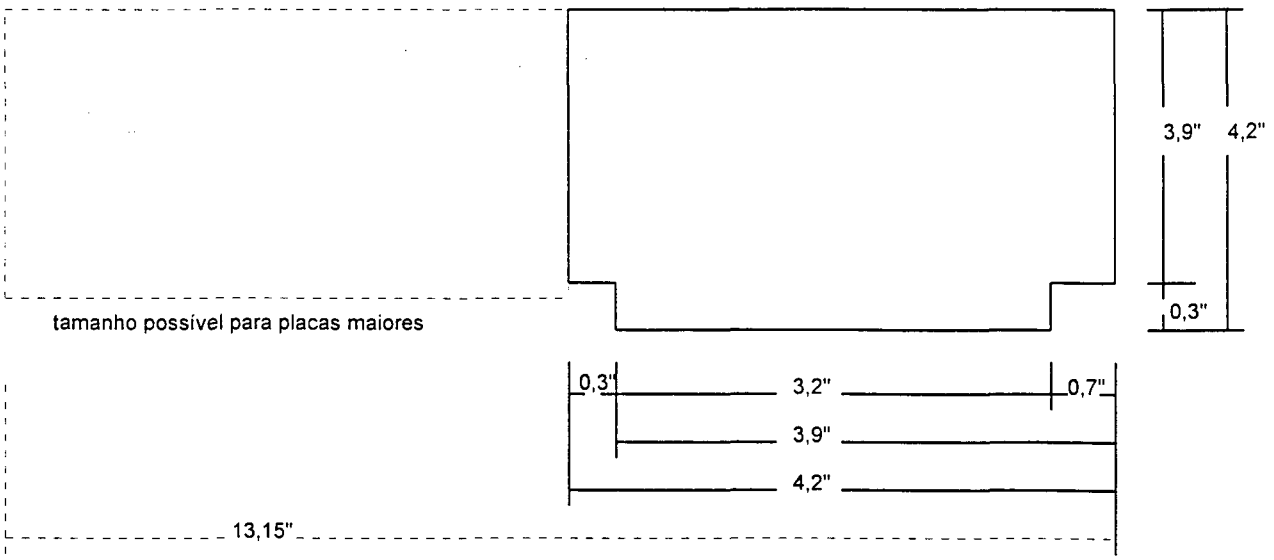


Figura B.4. Dimensões da placa

B.5. ANÁLISE DOS RESULTADOS APRESENTADOS POR ESTA INTERFACE

Esta placa foi construída e testada, oferecendo bons resultados para aplicações no ensino e no apoio de sistemas automáticos.

Este projeto apresentou um placa, com um circuito extremamente simples, inclusive sem um circuito para atendimento de interrupções, porém o usuário poderá fazer isso, caso seja necessário. A figura B.5, sugere um circuito para atendimento de interrupções.

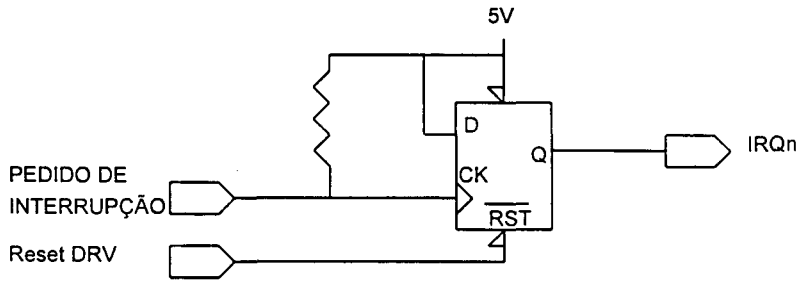


Figura B.5. Circuito para atendimento à interrupção.

A borda de subida da linha "pedido de interrupção", ajusta o **latch**, que por sua vez ativa a linha IRQ do barramento do PC. O **latch** mantém o pedido ativo até que receba o

segundo pulso INTA enviado pela CPU, desta forma, ele pode ser "zerado" por uma instrução OUT enviada pela rotina de atendimento à interrupção.

9. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Ammeraal, Leendert. "Programming Principles in Computer Graphics", John Wiley & Sons, 1987.

[2] Asser, Stuart M. at alli, "Microcomputer Theory and Servicing", Merril, 1993, second edition.

[3] Ayers, Kenneth E. "An Object-Oriented Logic Simulator", Dr. Dobb's Journal, December 1989, pp. 72-78.

[4] Berry, John. "Programando em C++", Makron Books do Brasil, 1991.

[5] Boddendorf F. "Computer in der Fachlichen und Universitären Ausbildung", Oldenbourg, Germany, 1990.

[6] Bonatti, Ivanil & Madureira, Marcos. "Introdução á Análise de Circuitos Lógicos", Editora Unicamp, Brasil, 1990.

[7] Booch, Grady, "Object Oriented Design With Applications", Benjamim/Cummings Publishing Company Inc., California, 1991.

[8] CACI Products Company, "A Quick Look at MODSIM II - The Object-Oriented Language", catálogo do produto,1989.

[9] CACI Products Company, "A Quick Look at SIMSCRIPT II.5 With SIMGRAPHICS", Catálogo do produto,1989.

[10] Castro, Claudio de M. "Estrutura e Apresentação de Publicações Científicas", McGraw Hill do Brasil, 1976.

[11] Coad, Peter & Yordan, Edward. "Object-Oriented Analysis", Second Edition, Prentice Hall, 1991.

[12] Concern Standardization Departament "LOGIC SYMBOLS - Manual for the Aplications", Publication UT-D 1164/1 (10), november 1977.

[13] Cutler, Michal & Eckert, Richard R. "A Microprogrammed Computer Simulator", IEEE Transaction on Education, vol. E-30, nº 3, pp. 135-141,August 1987.

[14] Daghlian, Jacob. "Lógica e Álgebra de Boole", Editora Atlas, Brasil, 1986.

[15] Davies, Ruth M. & O'Keefe, Robert M., "Simulatrion Modeling with Pascal", Prentice Hall,UK,1989.

[16] Diab, H.B. & Demashkieh I. "A Computer-Aided Teaching Package for Microprocessor Systems Education", IEEE Transaction on Education, vol. 34, nº 2, pp. 179-183, May 1991.

[17] E+PK Ingeniewreüro, "FlowLearn", Aacheu, RFA, 1991.

[18] Embley, David W. at alli, "Object-Oriented System Analysis - A Model-Driven Approach", Yordan Press, New Jersey, 1992.

[19] Farley, Richard, "Software Enginnergin Concepts", Mc Graw-Hill, 1984

[20] Fayek, Abdel-Moaty & CACI Products Company, "Introduction to Combined Discrete-Continous Simulation Using SIMSCRIPT II.5", Computer Science Department of California State University, Chico,1988.

[21] Fuchs, Kent W. et alli, "Workstation-Based Logic Animation and Microarchitecture Emulation for Teaching Introduction to Computer", IEEE Transaction on Education, vol 32, nº 3, pp. 218-225, August 1989.

[22] Furth, Borivoje. "An Advanced Laboratory for Microprocessor Interfacing and Communication", IEEE Transaction on Education, vol 32, nº 2, pp. 124-128, May 1989.

[23] Gibson, Glenn A. "Computer Systems: Concepts & Design", Prentice Hall, 1991.

[24] Golden, Donald G., "Software Engineering Considerations for the design of Simulation Languages", Simulation, pp. 169-178, october 1985.

[25] Graf Elektronik Systeme, "LogSim - Profilog", Kemptem, RFA, 1990

[26] Graf, Gerd, "Der PC Zwischen Analog un Digital", MC, pp. 68-73, juli 1990.

[27] Hall, Douglas V. "Microprocessors and Interfacing: Programming and Hardware", McGraw Hill, USA, 1985.

[28] Jones, Douglas W. "A Minimal CISC", Computer Architecture News, acmPRESS, New York, USA, June 1988.

[29] Kienle, Steven. "Network Graphs in Object Pascal", Dr. Dobb's Journal, pp. 17-25, December 1989.

[30] Kohavi, Zvi "Switching and Finite Automata Theory", TAT McGraw Hill Publishing Co. Ltd., India, 1978.

[31] Kraus Jr, W. "Aspectos do Planejamento Curricular e da Atividade de Ensino em Engenharia de Controle e Automação", Dissertação de mestrado, CPGEEL, UFSC, setembro de 1991.

[32] LABTECH, "Labtech Notebook & Labtech Control", Catálogo do fabricante.

[33] Lücke, Hermann A. H. "Computadores Pessoais Para Automatização de Laboratórios", IV Congresso Nacional de Automação Industrial (CONAI), São Paulo-SP 1990.

[34] Lücke, Hermann. A. H. "Um Laboratório Computadorizado para Disciplina de Hardware da Área de Computação", Projeto interno do Departamento de Ciências Estatísticas e da Computação da UFSC.

[35] Lücke, Hermann. & A. H., Negri & V. Madeira, M. "Programação Orientada a Objetos na Automação Industrial", CONAI, 1994.

[36] Malvino, Albert Paul, "Microcomputadores e microprocessadores", São Paulo, McGraw Hill do Brasil, 1985.

[37] Mann, Daniel. "The Universal Debugger Interface", Dr. Dobb's Journal, pp. 58-68, September 1992.

[38] McLaughlin, Michael P. "Simulated Annealing", Dr. Dobb's Journal, pp. 26-37, September 1989.

[39] Meirelles, Fernando S. "Informática: Novas aplicações com Microcomputadores", McGraw Hill, Brasil, 1988.

[40] Mendelson, Elliot. "Álgebra Booleana e Circuitos de Chaveamento", Schaum-McGraw Hill, 1977.

[41] Moreira, Mércia. "Concepções Psicológicas do Desenvolvimento e Aprendizado que Permeiam as Práticas Pedagógicas: Pressuposto Epistomológicos Básicos nas Relações Sujeito-Objeto", UFMG, Reprodução feita pelo CEC/CTC/UFSC.

[42] Nevin, Randy. "Autorouting with the A* Algorithm", Dr. Dobb's Journal, pp. 16-86, September 1989.

[43] Nolan, Tom. "Real-Time Data Acquisition Using DMA", Dr. Dobb's Journal, pp. 28-37, January 1990.

[44] Oppenheim, Alan V. & Willsky, Alan, "Signals and Systems", Prentice/Hall International Inc., 1983.

[45] OrCAD System Corporation, "User's Guide of OrCAD/VST", Oregon, USA, 1989.

[46] Pegden, C.D., at alli, "Introduction to Simulation Using SIMAN", McGraw Hill, USA, 1990.

[47] Pressman, Roger, "Software Engineering: A Practitioner's Approach", Mc Graw-Hill, 1987

[48] Purvis, Richard E. et alli, "MIME: An Educational Microprogrammable Minicomputer Emulator", IEEE Transaction on Education, vol E-24, nº 4, pp. 257-2262, november 1981.

[49] Roden, Thomas. "High_Resolution Timing", Dr. Dobb's Journal, pp. 42-48, September 1992.

[50] Rumbaugh, J. at alli, "Object-Oriented Modeling and Design", Prentice Hall, New Jersey, 1991.

[51] Schaufelberger, Walter. "Design and Implementation of software for Control Education", IEEE Transaction on Education, vol 33, nº 3, pp. 291-297, Aug. 1990.

[52] Shimizu, Tamio, "Simulação em Computador Digital", São Paulo, Ed. da Universidade de São Paulo, 1975

[53] Silvester, Peter. "Introducing Computer Structure by Machine Simulation", IEEE Transaction on Education, vol 33, nº 4, pp. 326-332, Nov. 1990.

[54] Smith, Michael R. "A Microprogrammable Microprocessor Simulator and Development System", IEE Transaction on Education, vol. E-27, nº 2, pp. 93-100, May 1984.

[55] Souza, Janice T. P. de & Neves, Luiz C. A. "Manual de Orientação Básica para Elaboração de Monografia", Universidade Estadual de Maringá.

[56] Taub, Herbert & Schiling, Donald. "Digital Integrated Electronics", McGraw Hill, 1977.

[57] Taub, Herbert. "Circuitos Digitais e Microprocessadores", McGraw Hill do Brasil, 1982.

[58] Tavernini, Lucio, "A User-Friendly Interactive Turbo Pascal Simulation Toolkit", Simulation, pp. 45-55, august 1989.

[59] Wellstead, Peter. "Teaching Control with Laboratory Scale Models", IEEE Transaction on Education, vol 33, nº 3, pp. 285-290, August 1990.

[60] Wendt, Siegfried, "Nichtphysikalische Grundlagen der Informationstechinik - Interpretierte Formalismen", Springer-Verlag, Germany, 1989.

[61] Wolfe, Steve, "PC-Based Test Bech", Radio Eletronics, pp. 39-48, June 1992.

[62] Yen, Ruey-Fong & Kim, Yongmin. "Development and Implamentation of an Educational Simulator Software Package for a Specific Microprogramming Architeture", IEEE Transaction on Education, vol E-29, nº 1, pp. 1-11, February 1986.

[63] Pacote de manuais do Microsoft C/C++ Compiler V 7.00, Microsoft Corporation, 1991.